ŠOLSKI CENTER VELENJE
ELEKTRO IN RAČUNALNIŠKA ŠOLA
Trg mladosti 3, 3320 Velenje
MLADI RAZISKOVALCI ZA RAZVOJ ŠALEŠKE DOLINE

RAZISKOVALNA NALOGA

OBOGATENA VR-IZKUŠNJA

Tematsko področje: APLIKATIVNI INOVACIJSKI PREDLOGI IN PROJEKTI

Avtorja:
Tim Andrejc, 3. letnik
Aneja Pirnat, 3. letnik

Mentorja:
Rok Urbanc, dipl. inž.
Samo Železnik, inž.

Velenje, 2022

Raziskovalna naloga je bila opravljena na Šolskem centru Velenje, Elektro in računalniški šoli.

Mentorja: Rok Urbanc, dipl. inž.
Samo Železnik, inž.

Datum predstavitve: marec 2022

KLJUČNA DOKUMENTACIJSKA INFORMACIJA
ŠD Šolski center Velenje, Elektro in računalniška šola, šolsko leto 2021/2022
KG motorčki / odziv / vonj / imerzija
AV ANDREJC, Tim / PIRNAT, Aneja
SA URBANC, Rok / ŽELEZNIK, Samo
KZ 3320 Velenje, SLO, Trg mladosti 3
ZA ŠCV, Elektro in računalniška šola
LI 2022
IN **OBOGATENA VR IZKUŠNJA**
TD Raziskovalna naloga
OP VII, 58 str., 6 pregl., 34 sl., 5 graf., 7 pril., 19 vir.
IJ SL
JI sl / en
AI V današnjem svetu sta zabava in sprostitev vedno bolj pomembni in vedno znova iščemo izhod iz našega vsakdanjega življenja. Nekateri pobegnejo v virtualno resničnost, zato je bil najin cilj v tej raziskovalni nalogi odkriti, kako bi bila uporaba virtualne resničnosti bolj realistična in poglobljena. To sva storila s prikazom haptične odzivnosti telesa na tresljaje in vonjave. Naredila sva igro, ki omogoča zabavno in bolj poglobljeno VR-doživetje. Haptično odzivnost sva dosegla z motorčki, pritrjenimi na jopič uporabnika, ki povzročajo tresljaje in mu tako omogočajo, da čuti elemente igre na telesu fizično. Dodala sva tudi izkušnjo z vonjavami, pri kateri uporabnik s pomočjo dejanske arome eteričnega olja v prostoru podoživlja vonj v navidezni resničnosti igre. Igro tako zaznava s čuti in to njegovo doživljanje sicer fiktivne realnosti postane intenzivnejše in s tem bolj sproščujoče.

KEY WORDS DOCUMENTATION
ND Šolski center Velenje, Elektro in računalniška šola, school year 2021/2022
CX motors / response / smell / immersion
AU ANDREJC, Tim / PIRNAT, Aneja
AA URBANC, Rok / ŽELEZNIK, Samo
PP 3320 Velenje, SLO, Trg mladosti 3
PB ŠCV, Elektro in računalniška šola
PY 2022
TI **ENHANCED VR EXPERIENCE**
DT RESEARCH WORK
NO VII, 58 p., 6 tab., 34 fig., 5 graf., 7 ann., 19 ref.
LA SL
AL sl / en
AB Nowadays entertainment and relaxation are becoming more and more important and we are constantly in search of finding additional ways to escape our daily lives. Some take refuge in virtual reality, therefore our goal in this research project was to discover how the use of virtual reality could become more realistic and a true in-depth experience. We did this by showing the body's haptic response to vibrations and odours. We created a game that enables a fun and more in-depth VR adventure. We achieved haptic responsiveness by using motors attached to the user's jacket, which cause vibrations, thus allowing the individual to feel the elements of the game physically on the body. In addition, we enriched the game experience with scents by making use of the actual aroma of the essential oils provided in the room to create a more vivid occurrence in the game. As a result, users perceive the game with their senses and their experience of an otherwise fictitious reality becomes more intense and consequently more relaxing.

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

KAZALO GRAFIKONOV

1. UVOD

V današnjem svetu sta zabava in sprostitev vedno bolj pomembni in pogosto iščemo nove izhode iz našega vsakdanjega življenja. Nekateri to storijo s pobegom v virtualno resničnost, zato je bil v tej raziskovalni nalogi najin cilj odkriti, kako bi bila uporaba virtualne resničnosti za uporabnika bolj realistična in poglobljena. To sva storila z uporabo haptične odzivnosti na telesu in s pomočjo vonjav, ki spodbudijo uporabnikov odziv. Naredila sva igro, ki dobro prikaže zabavno in bolj poglobljeno VR-doživetje. Haptično odzivnost sva dosegla z motorčki, ki vibrirajo in so pritrjeni na jopič, ki si ga uporabnik nadene in tako čuti elemente igre na telesu, saj motorčki zavibrirajo. Dodala sva tudi izkušnjo z vonjavami, pri kateri uporabnik vonja »okolico v igri« s pomočjo eteričnih olj. To doda najinemu projektu še večji realizem, saj se uporabnik v igro poglobi skoraj z vsemi čuti. Idejo za jopič sva dobila pri opazovanju vibrajočih motorčkov, ki se nahajajo v raznih kontrolorjih, kot npr. v igralnih konzolah Playstation, Xbox, in tudi VR Viveu. Ti pri igranju spodbujajo večjo odzivnost in tudi boljši občutek, zato sva jih hotela uporabiti še na drugačen način – na jopiču. V izkušnjo sva hotela vključiti tudi druga čutila in sva se odločila za vonj. To sva storila z ventilatorji, ki se nahajajo na različnih delih sobe, v kateri igralec igra. Pred ventilatorji stoji vonjava, ki jo nato ventilator ponese po sobi do igralca, ta pa lahko s pomočjo tega »vonja okolico« v igri.

2. HIPOTEZE

1. Vonjave in haptični jopič bodo imele na uspešnost igralca v igri pozitiven učinek.
2. Vonjave in haptični jopič bodo imele na subjektivno izkušnjo igralca pozitiven učinek.
3. Učinek vonjav bo imel bolj pozitiven učinek na uspešnost igralca v igri kot haptični jopič.
4. Izkušnja z vonjavami bo imela bolj pozitiven učinek na subjektivno izkušnjo igralca kot izkušnja s haptičnim jopičem.

3. RAZLAGA POJMOV

Unity – program, ki omogoča ustvarjanje iger na različnih platformah.
VR – virtualna resničnost, do katere dostopamo preko računalnika in posebne VR opreme (očali, senzorji, posebnimi kontrolerji …).
C# – programski jezik.
Python – programski jezik.
Asset – objekt, ki ga uporabljamo pri oblikovanju projektov v Unityju.
SDK – (ang. software developer kit) komplet za razvoj programske opreme.
Websocket – računalniški komunikacijski protokol.
Raspberry Pi – mikroračunalnik.
V – volt.
Arduino – mikrokrmilnik.
Json – standardna oblika zapisa za izmenjavo podatkov.
mm – milimeter.
HMD – angl. head mounted display
Hz – herc

4. PREGLED OBJAV

4.1. ZGODOVINA VR IN POMEMBNEJŠI MEJNIKI

1956
Kinematograf Morton Helig je izumil *Sensoramo,* ki je bila prvi VR-stroj. *Sensorama* je bila velika kabina, v katero so lahko šli štirje ljudje. Uporabljala je več različnih vrst tehnologije za stimulacijo več čutov. Vsebovala je barvni 3D-video, zvok, vibracije, vonj in efekte za atmosfero, kot na primer veter. Helig je verjel, da je to prihodnost filma in za *Sensoramo* je bilo posnetih šest kratkih filmov. [1]

1960
Helig je izumil in patentiral telesfersko masko. Bila je prvi naglavni zaslon ali HMD, kar je omogočalo gledanje 3D-slik s širokim zornim kotom in zvokom, ni pa še omogočala sledenja gibanja. [2]

1968
Sutherland je s svojim študentom Bobom Sproullom ustvaril prvi HMD, ki je podpiral virtualno realnost. Poimenovala sta ga *The Sword of Damocles.* Ta naprava je bila povezana na računalnik, vendar je bila dokaj enostavna, saj je lahko prikazovala le oblike. Perspektiva oblik se je spreminjala tako, da je uporabnik premikal glavo. *The Sword of Damocles* nikoli niso dokončno dodelali, in sicer zaradi njegove prevelike teže. [3]

1991
Istega leta je skupina *Virtuality group* izdala svojo napravo *Virtuality*. Naprava je omogočala igralcem računalniških igric, da so lahko igranje izkusili v 3D-okolju. To je bil tudi prvi masovno proizveden VR-sistem. [4]

4.2. KAJ JE HAPTIKA?

Haptična tehnologija, znana tudi kot kinestetična komunikacija ali 3D-dotik, se nanaša na katerokoli tehnologijo, ki lahko ustvari izkušnjo dotika z uporabo sil, vibracij ali gibov na uporabnika. [5]

4.3. ZGODOVINA HAPTIKE

1970
Izdelovalci arkadnih iger so začeli razmišljati, kako bi izkušnjo arkadnih iger za uporabnika naredili bolj zanimivo. Leta 1976 je bila igra *Fonz*, ki so jo ustvarili pri podjetju SEGA, prva, ki je imela vibrirajoče povratne informacije. Igra je omogočala uporabniku, da je začutil vibracijo, ko se je v zaletel v nasprotnika. Ta funkcija je bila igralcem všeč, zato so v drugih podjetjih začeli razmišljati, kako bi lahko enake funkcije ustvarili na svojih arkadnih igrah. [6]

1980
Leta 1983 je bila izdana dirkaška arkadna igra, imenovana *TX-1*. Izdelalo jo je podjetje Tatsumi. Uporabljala je pedalke, ki so imele vibracijsko odzivnost. To je dalo igralcem občutek, da vozijo avto. [7]

1990
Izdelovalci igralnih kontrolorjev so pri arkadnih igrah opazili napredek haptične odzivnosti in jo začeli uvajati v svoje kontrolorje.

Leta 1997 je Nintendo izdal *N64 Rumble Pack* s funkcijo vibrirajoče odzivnosti. [8] Kasneje istega leta je Sony izdal *DualShock* kontroler za *Playstation.* [9]

Leta 1994 je bil izdan *Interactive Vest*, ki ga je ustvarilo podjetje Aura Systems. To je bila naprava, ki jo je lahko uporabnik nosil na sebi in je pretvarjala avdio signale v vibracije, da je lahko med igranjem čutil udarec. Podjetje je prodalo več kot
400 000 naprav, ki so se prodajale na trgu za 99 dolarjev. [10]

Zatem se je tehnologija haptične odzivnosti nehala razvijati za približno desetletje.

Slika 1: Interactive Vest (viri slik: vir[1])

2000

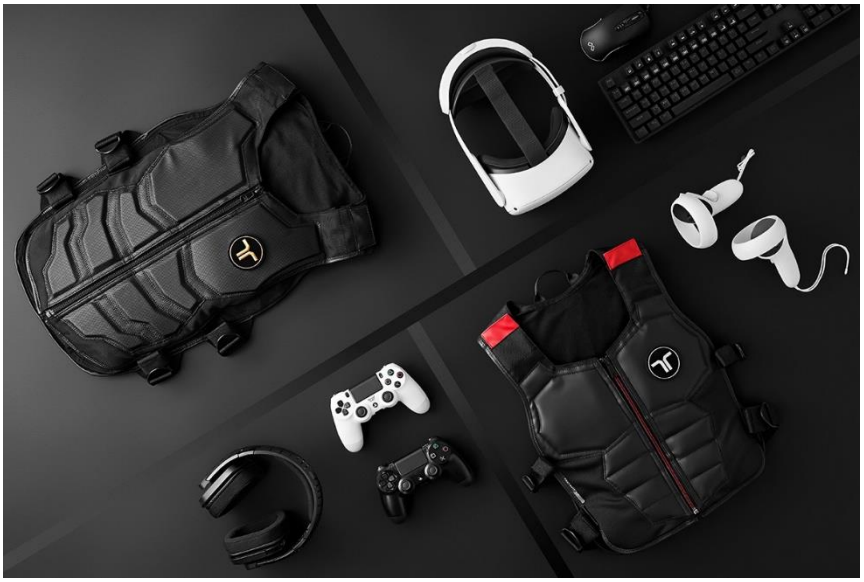
Leta 2007 je bil izdan *ForceWear Vest* na konferenci *Game Developers* v San Franciscu. Podjetje TNGames je spremenilo ime v *3D Space Vest* novembra istega leta. *Space Vest* je bil za razliko od prejšnjih haptično odzivnih jopičev usmerjen, kar je pomenilo, da je lahko igralec čutil dogajanje tudi izven svojega vidnega polja. *Space Vest* je podpiral skoraj 50 igric. [11]

4.4. BHAPTICS

bHaptics je podjetje, ki se trenutno s haptično odzivnimi oblačili ukvarja najbolj. Z njihovimi odkritji na področju haptike sva si pri izdelavi raziskovalne naloge pomagala tudi midva. Na trgu imajo več različnih haptično odzivnih oblačil, med njimi dva različna jopiča, haptične rokavice in nožno ter naglavno haptiko. Njihovi izdelki podpirajo zelo veliko različnih video igric tako na VR-u kot na računalniku. [12]

4.4.1. BHAPTICS PLAYER

Njihov bHaptics Player je vmesnik, ki omogoča tudi »mod support«, s pomočjo katerega lahko uporabniki dodajo haptično odzivnost na katero izmed iger, ki jih bHaptics še ne podpira, s pomočjo modiranja. Prav tako ima že v naprej narejene mode za veliko iger.



Slika 2: bHaptics izdelki (viri slik: vir[2])

4.5. KAJ JE MOD IN KAJ JE MODIRANJE?

Modiranje je proces spreminjanja video iger s strani igralcev ali oboževalcev. Lahko spreminjamo več vidikov video igre, na primer, kako izgleda, kako se obnaša, dodamo kakšne nove funkcije … Mod je program, ki nastane z modiranjem.

4.6. SMELL-O-VISION

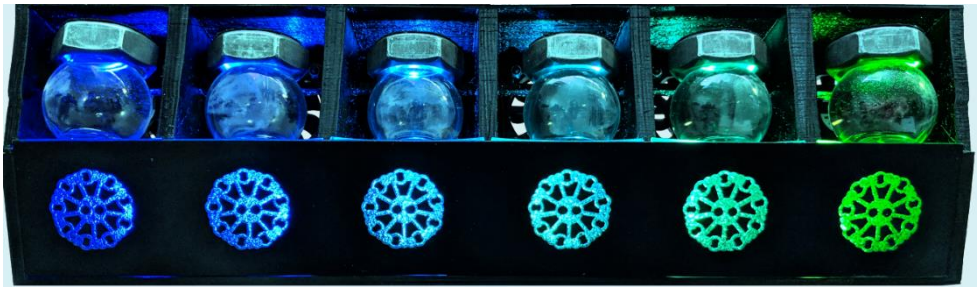
4.6.1. SCENT OF MYSTERY

Leta 1960 je bil narejen film *Scent of Mystery*, pri katerem so prvič uporabili novo tehnologijo, imenovano Smell-O-Vision. V gledališki dvorani so se ob dogodkih v filmu sproščale skozi odprtine pod sedeži različne arome. Ob filmu *Scent of Mystery* so bile uporabljene arome kave, tobaka, bencina, banan in čistila za čevlje. Publiki je bilo zanimivo, vendar so kritiki to tehnologijo zavračali.
Izkušnja s haptiko in tehnologijo Smell-O-Vision je imela veliko tehničnih težav in je bila tudi do neke mere moteča za gledalce. Nekateri so menili, da je čudna in da jih preveč odvrne od filma. [13]

Po Smell-O-Visionu se tovrstna tehnologija ni več razvijala.

4.6.2. CILIA DEVELOPER KIT

Podjetje Hapticsol je ustvarilo *Cilia Developer Kit*, v katerega je vključena naprava, imenovana *Cilia*, ki ob dogodkih v igrah sprošča arome. Vključuje arome, imenovane Waterfall, Fresh Air, Rocky plant, Egyptian Musk, Cappuccino in Fresh Mint. *Cilia* ima 6 poličk, na katere lahko postavimo arome, za njimi pa so mali ventilatorji, ki se ob dogodkih sprožijo in nato ugasnejo. Z njihovo idejo sva si pomagala pri najini raziskovalni nalogi in uporabila podoben model. Na njihovi spletni strani znaša cena naprave 399 dolarjev. *Cilia* je po ocenah kar dobro delujoča in igranju doda pozitiven in bolj realističen učinek. [14]



Slika 3: Cilia (viri slik: vir[3])

5. UPORABLJENA OPREMA

5.1. VIVE PRO

Za nastanek raziskovalne naloge sva uporabila Vive PRO. Vive so razvili v firmi HTC, cena na spletu pa znaša okrog 1300 EUR. V kompletu sta senzorja, ki zaznavata premikanje, VR-očala in dva kontrolorja. Ta VR sva uporabila, ker smo ga že imeli na naši šoli.

SPECIFIKACIJE VIVE PRO-ja: [15]

- zaslon: Dual AMOLED z diagonalo 3,5,
- ločljivost: 1440 x 1600 pikslov (2880 x 1600 pikslov skupaj),
- hitrost osveževanja: 90 Hz,
- polje: 110 stopinj,
- zvok: slušalke s certifikatom visoke ločljivosti (odstranljive),
- podpora za slušalke z visoko impedanco,
- vhod: vgrajeni mikrofoni,
- povezave: Bluetooth, vhod USB-C za zunanje naprave,
- senzorji: sledenje SteamVR, G-senzor, žiroskop, bližina, nastavitev udobja za oči. (IPD),
- ergonomija: razbremenitev oči s prilagoditvijo razdalje leč,
- nastavitev udobje za oči,
- nastavljive slušalke,
- nastavljiv naglavni trak.

Slika 4: Vive Pro (viri slik: vir [4])

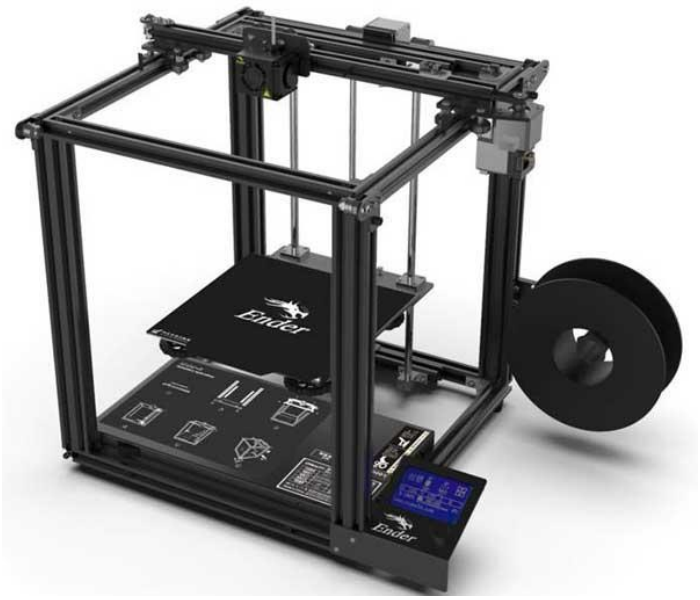
5.2. CREALITY ENDER 5 PRO 3D PRINTER

Za izdelavo najinih 3D-modelov sva uporabila Creality Ender 5pro 3Dprinter. Na spletu znaša njegova cena od okrog 290 do 370 EUR. Ta printer sva uporabila, ker smo ga že imeli na šoli.

SPECIFIKACIJE CREALITY ENDER 5 PRO-ja: [16]

- 3D-tiskalnik je nadgrajen tiskalnik Ender 5;
- kakovost tiskanja visoke natančnosti,
- premika se bolj gladko in omogoča natančnejšo tiskanje;
- ima vgrajen napajalnik, ki se v 5 minutah segreje na 110 °C;
- tiskalnik je z napajanjem zaščiten pred izpadi električne energije. Če se zgodi izpad elektrike, se lahko z zadnjim slojem tiskanje nadaljuje, kar prihrani čas in zmanjša odpadke;

- velikost tiskalne površine: 220 mm x 220 mm x 300 mm,
- hitrost tiskanja: 30–150 mm/s,
- debelina sloja: 0.06–0.3 mm,
- tiskalna glava: enojna,
- zaslon: visokokakovostni grafični LCD-zaslon 128 x 64 px,
- premer filamenta: 1.75 mm,
- temperatura grelne mize: max. 110 °C,
- temperatura ekstrudorja: max. 255 °C,
- povezava: USB-kartica, SD-kartica,
- potrebna moč: 230 V,
- programska oprema za nadzor: Cura, Slic3r, Repetier-host.



Slika 5: Creality Ender 5 PRO (viri slik: vir [5])

5.3. RASPBERRY PI 4

Raspberry Pi 4 ima povečano hitrost procesorja, kar omogoča veliko zmogljivost. Ima boljši pomnilnik in boljšo povezljivost v primerjavi s prejšnjo generacijo, vendar ohranja enako porabo energije. To je eden izmed razlogov najine odločitve, da ga uporabiva pri raziskovalni nalogi. Raspberry Pi 4 ponuja zmogljivost, ki je primerljiva z računalniškimi sistemi x86. Glavne značilnosti vključujejo visoko zmogljiv 64-bitni štirijedrni procesor, podporo za dvojni zaslon pri ločljivostih do 4K prek mikro-HDMI, dvopasovni 2,4/5,0 GHz brezžični LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0 in PoE zmogljivost (prek ločenega dodatka PoE HAT). [17]

Slika 6: Raspberry Pi model 4 (viri slik: vir [6])

5.4. ARDUINO UNO

Arduino UNO je poceni, prilagodljiv in enostaven za uporabo. Je odprtokodna mikrokrmilniška plošča, ki jo je mogoče integrirati v različne projekte. To ploščo je mogoče povezati z drugimi ploščami Arduino, Arduino Shield, Raspberry Pi … Nanjo lahko priključimo releje, LED-diode, motorje itd. Arduino UNO ima AVR mikrokrmilnik Atmega328, 6 analognih vhodnih nožic in 14 digitalnih I/O zatičev, od katerih se 6 uporablja kot PWM-izhod. Ta plošča vsebuje vmesnik USB, tj. USB-kabel se uporablja za povezavo plošče z računalnikom, programska oprema Arduino IDE (integrirano razvojno okolje) pa se uporablja za programiranje mikrokrmilnika. Enota ima 32 KB velik pomnilnik, ki se uporablja za shranjevanje, medtem ko ima SRAM 2 KB prostora in EEPROM 1 KB prostora. Deluje na napetosti 5 V, medtem ko se vhodna napetost giblje med 6 in 20 V. Priporočena vhodna napetost je med 7 in 12 V. [18]

Slika 7: Arduino (viri slik: vir [7])

6. ZAČETKI RAZISKOVANJA

Projekt sva začela s testiranjem playerja bHaptics. Uporabila sva že narejen mod za strelsko igro CS:GO, da sva videla, kako deluje. Mod sva spremenila, da je izpisalo, kaj se je zgodilo v igri npr. 'Goriš', 'Umrl si' itd. Namen te spremembe je bil videti, kaj program pošilja in bolje spoznati jezik Python, saj ga prej nisva veliko uporabljala. Na začetku sva podatke pošiljala med dvema računalnikoma na mesto med računalnikom in Raspberry Pijem, da sva lažje videla, kakšni podatki se pošiljalo in kako deluje povezava med strežnikom in odjemalcem.
Oba računalnika sva povezala na isti IP in port ter s tem vzpostavila povezavo.

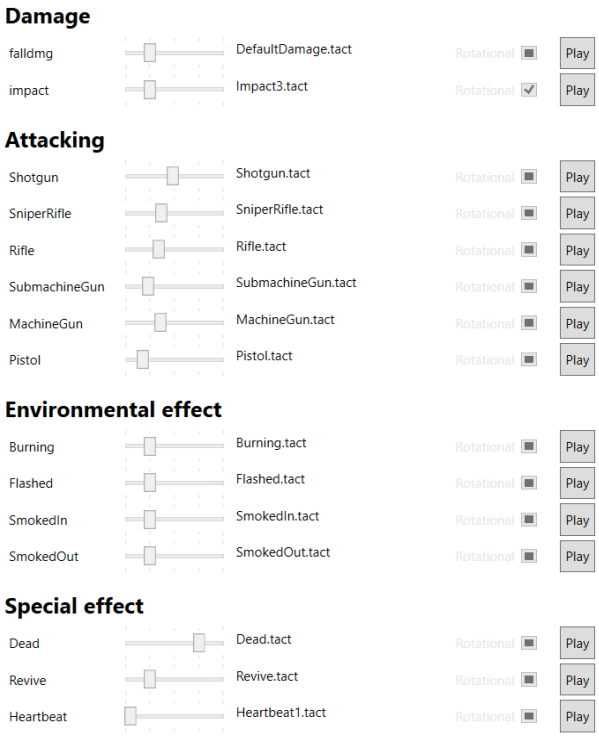
```
s = socket.socket()
ws = create_connection("ws://localhost:15881/v2/feedbacks")
def povezava(s):
    host = '172.20.10.5'
    port = 42424
    s.connect((host, port))
povezava(s)

async def server(websocket, path):
    while True:
        try:
            data = await websocket.recv()
            data = json.loads(data.replace("'", "\""))
            status = str(data)
            izpis = ""
            if 'falldmg' in status:
                izpis = "Padel si"
            elif 'impact' in status:
                izpis = "Zadel si nekaj"
            elif 'Dead' in status:
                izpis = "Umrl si"
            elif 'Burning' in status:
                izpis = "Goris"
            elif 'Win' in status:
                izpis = "Zmagal si"
            elif 'Reload' in status:
                izpis = "reload"
            elif 'ChangeWeapon' in status:
                izpis = "Spremenil si orozje"
            elif 'BombExploded' in status:
                izpis = "Bomba je eksplodirala"
            s.send(izpis.encode())
        except Exception as e:
            print(e)

start_server = websockets.serve(server, "localhost", 15881)
asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

Slika 8: Slika CS:GO, moda kode in strežnika

```
import socket

def Main():
    host = '172.20.10.5'
    port = 42424
    s = socket.socket()
    s.bind((host, port))

    s.listen(1)
    c, addr = s.accept()
    while True:
        data = c.recv(1024).decode()
        print(data)
        if not data:
            break
        data = str(data).upper()
    c.close()
if __name__ == '__main__':
    Main()
```

Slika 9: Slika kode odjemalca

Po tem, ko sva uspešno povezala računalnika in zagnala igro, se je za vsak zapisan dogodek v igri tudi izpisalo sporočilo na ekranu odjemalca, kar je pomenilo, da je najin program deloval. Naslednji korak je bil, da poveževa strežnik z Raspberry Pijem na mesto z dvema računalnikoma in da se ob določenih dogodkih prižge na njem določen pin, ki sproži motorčke.



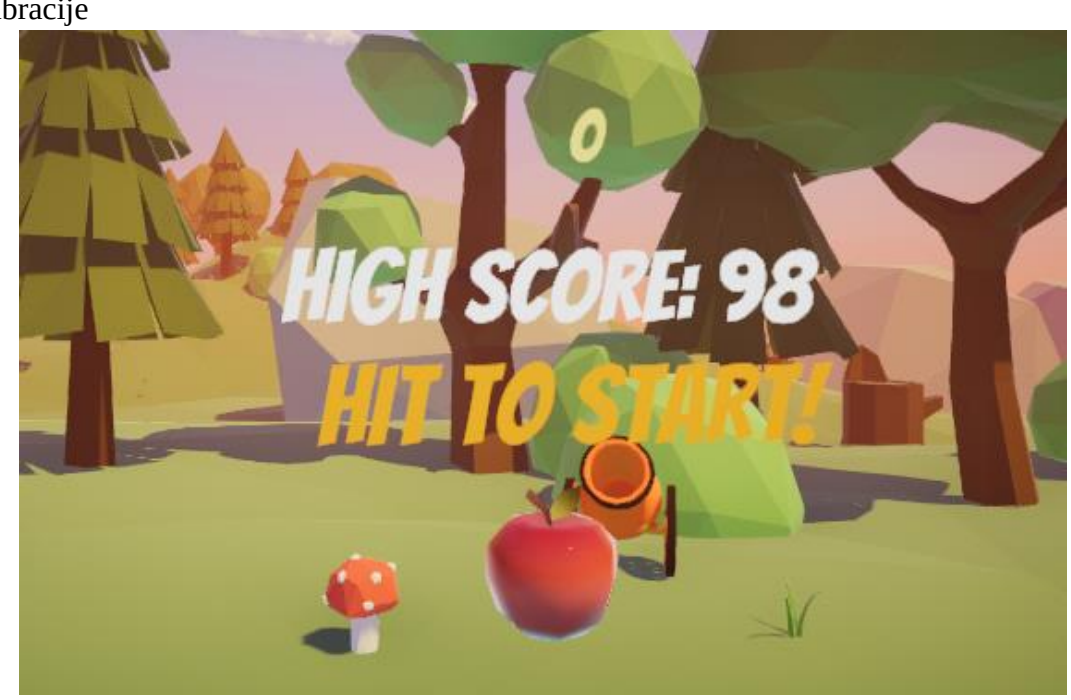
Slika 10: Slika moda CS:GO

Za ta namen sva spremenila kodo odjemalca, da je delovala na Raspberry Piju. Ko sva to storila, sva lahko preizkusila, ali motorčki vibrirajo v CS:GO ali ne. Zagnala sva igro in mod na playerju bHaptics in ugotovila, da najin program deluje. Motorčki so v igri vibrirali ob dejanjih, ki so bili določeni v modu, s katerim sva jim lahko določila tudi jakost.

Naslednji korak je bil, da izdelava svojo igro, s ciljem, da narediva enostavno, vendar zabavno igro, ki bo vključila vse dele najinega projekta.

7. IZDELAVA IGRE

Izdelave igre sva se lotila v okolju Unity. Odločila sva se za igro, v kateri uporabnik v VR-ju stoji v gozdu. Ko se v igri pojavi, ima pred seboj sliko jabolka, nad katerim piše »HIT TO START« oz. »UDARI ZA ZAČETEK« in pa »HIGHSCORE« oz. »NAJVIŠJI DOSEŽEN REZULTAT«. Ko prereže jabolko, se igra začne. V uporabnika iz topov leti sadje, ki ga mora nato prerezati. Ko sadje prereže, na kontrolorju, ki ga drži v rokah, čuti vibracije

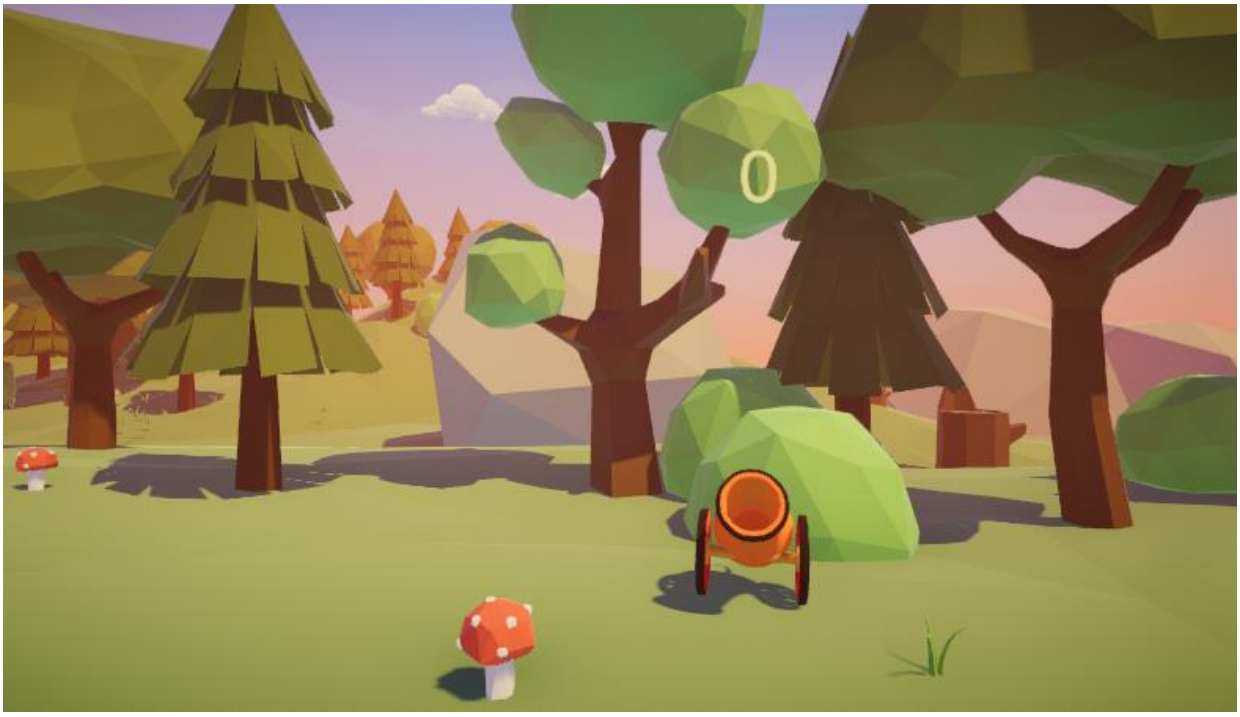


Slika 11: Slika začetka igre

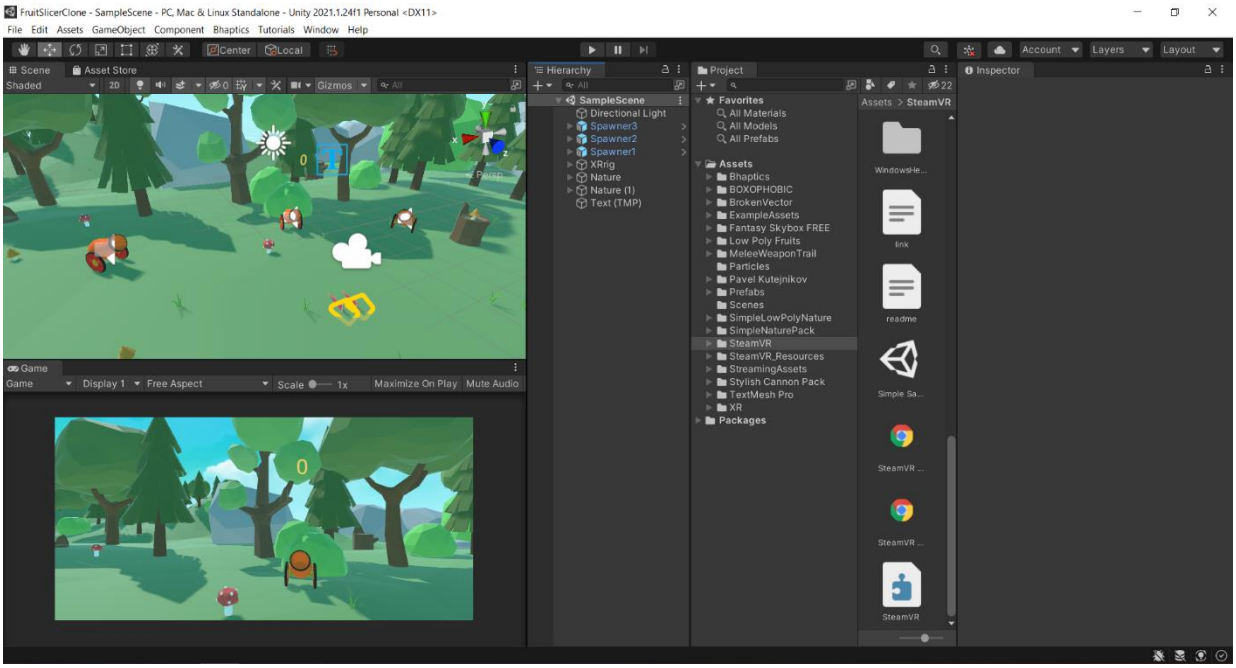
Vsakih petnajst sekund se spremeni smer, iz katere streljajo topovi, kar pomeni, da se mora igralec veliko obračati. Na obrat ga opozori časovnik, ki je v igro implementiran s haptičnim jopičem. Ko se čas za obrat bliža, uporabnik na sebi čuti vibracije odštevanja in ve, kdaj se mora obrniti. Kljub temu ima na ekranu še vizualno odštevanje, da ga na bližajoči obrat še dodatno opomni.
Na vsaki strani sobe je en ventilator, kar pomeni, da igralec vonja sadje iz katerekoli smeri.

Uporabila sva zastonj assete iz Unityjeve trgovine, ki sva jih nato uporabila za oblikovanje stopenj.

Unity omogoča dokaj enostavno vizualno ustvarjanje stopenj. Naložene assete uvozimo v naš projekt in jih nato po svoji volji postavljamo kjerkoli po praznem prostoru. Obračamo in premikamo jih s pomočjo x, y, z osi. Tako sva ustvarila videz igre, ki izgleda tako.



Slika 12: Izgled gozdne stopnje



Slika 13: Izgled Unityja

Tako izgleda Unity. Omogoča več različnih postavitev, vendar bova opisala tisto, ki sva jo uporabljala midva (imenovana je '2 by 3'). Zgornja leva slika nam nudi vpogled v urejanje stopnje, spodnja pa kako izgleda nekomu, ki je v igri. Desno imamo vse uvožene assete, ki so razporejeni po mapah za boljšo preglednost. Ko kliknemo na enega izmed njih, mu lahko spreminjamo lastnosti tudi na čisto desni strani ekrana. Lahko spreminjamo barvo, dodamo kakšne nove funkcije s pomočjo kode, spremenimo material predmeta … Ko smo s svojo igro zadovoljni, jo lahko zgoraj na sredini tudi zaženemo in preizkusimo. Če smo naredili kakšno napako, nas urejevalnik spodaj nanjo opozori, da jo lahko odpravimo.

Nato sva nadaljevala s kodo. Kode so napisane v programskem jeziku C#.

7.1. KODE ZA IGRO

7.1.1. KODA FRUIT SPAWNER

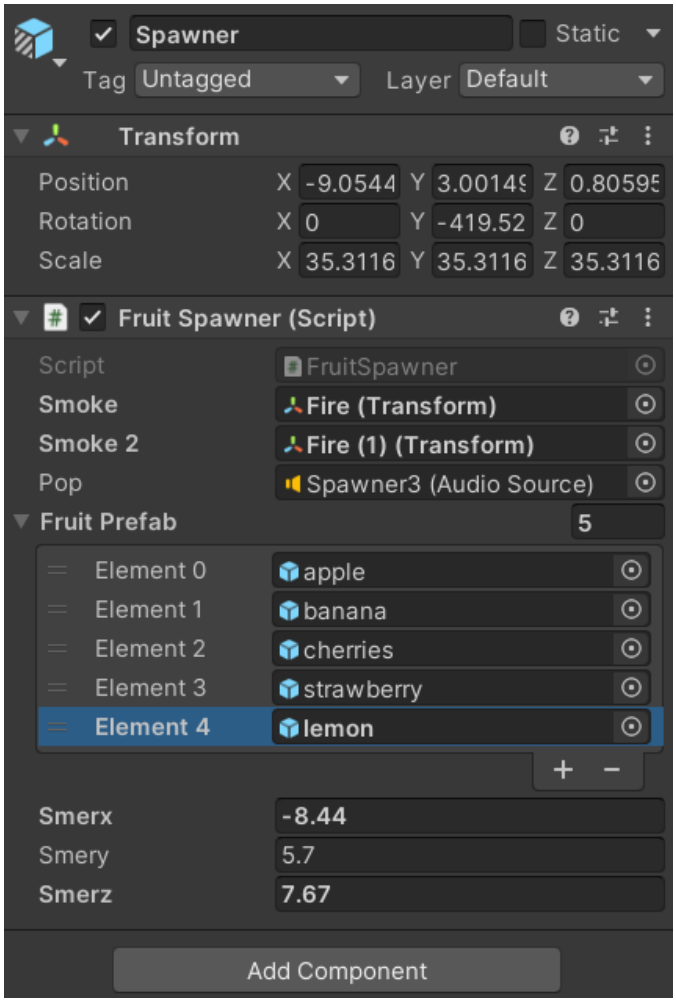
```
public void StartGame()
{
    StartCoroutine(Wait());
}
2 references
IEnumerator Wait()
{
    for (int t1 = 4; t1 > 0; t1--)
    {
        count.WaitTimer(t1);
        yield return new WaitForSeconds(1);
    }
    if (spawner1 == 0)
    {
        StartCoroutine(SpawnFruit());

    }
    else
    {
        drugi.ZacniDrugo();
    }
}
1 reference
public void ZacniPrvo()
{
    StartCoroutine(SpawnFruit());
}
1 reference
IEnumerator LetThereBeFruit()
{
    for (int t = 0; t <15 ; t++)
    {

        yield return new WaitForSeconds(1);
    }
    spawner1 = 1;
    StartCoroutine(Wait());
}
```

Slika 14: Prvi del kode Fruit Spawner

```
3 references
public IEnumerator SpawnFruit()
{
    int c = 0;
    spawner1 = 0;
    StartCoroutine(LetThereBeFruit());
    while (spawner1==0)
    {
        pop.Play();
        sadje = Random.Range(0, fruitPrefab.Length);
        GameObject go = Instantiate(fruitPrefab[sadje]);
        Rigidbody temp = go.GetComponent<Rigidbody>();

        if (c == 0)
        {

            smoke.GetComponent<ParticleSystem>().Play();
            c++;
        }
        else
        {

            smoke2.GetComponent<ParticleSystem>().Play();
            c = 0;
        }

        temp.velocity = new Vector3(Random.Range(smerx - 0.2f, smerx + 0.2f), Random.Range(smery - 0.2f, smery + 0.2f), Random.Range(smerz - 0.2f, smerz + 0.2f));
        temp.angularVelocity = new Vector3(Random.Range(-5f, 5f), Random.Range(-2f, 2f), Random.Range(-3.5f, 3.5f));
        temp.useGravity = true;

        Vector3 pos = transform.position;

        go.transform.position = pos;
        yield return new WaitForSeconds(Random.Range(0.5f, 2.5f));
    }
}
```

Slika 15: Drugi del kode Fruit Spawner

- S to kodo sva ustvarila element, ki ustvarja sadje in ga nato vrže v uporabnika. Poimenovala sva ga Fruit Spawner.
- »public void StartGame()« je funkcija, s katero se po prerezu začetnega jabolka začne 3-sekundni časovnik pred začetkom igre. Ko te tri sekunde potečejo, se začne glavna funkcija »SpawnFruit«, ki začne v uporabnika metati sadje. Hkrati začne tudi 15-sekundni časovnik. Ko ta časovnik poteče, sadje za 3 sekunde iz topov pred uporabnikom preneha leteti in nato začne leteti iz topov za uporabnikom.
- »public class Fruit Spawner : MonoBehaviour«, del kode Fruit Spawnerju določi elemente in njihove vrednosti. Elementi so AudioSource, element, ki nam omogoča dodajanje zvoka in določitev izvora zvoka, GameObject, element, ki nam omogoča dodajanje sadja, katerega Fruit Spawner zaluča v nas, in pa elementi Smerx, Smery, Smerz, s pomočjo katerih določimo smer, v kateri bo sadje letelo.
- Del kode Void Start() zažene funkcijo, ki ustvarja sadje.
- “pop.Play” predvaja zvok, vedno ko se pojavi novo sadje. Druga vrstica pod “pop.Play” naredi naključno vrsto sadja. Izbere izmed tistih, ki sva jih določila v Unityju. Tretja vrstica doda sadju fiziko.
- “temp.velocity” sadju doda zraven nastavljene vrednosti tudi malo variacije v smeri, v kateri bo letelo, “temp.angularVelocity” pa določi naključno rotacijo sadja. “temp.useGravity” da sadju gravitacijo. Z “yield return new WaitForSeconds” nato določimo naključno hitrost, v kateri bo Fruit Spawner v nas zalučal novo sadje. To se lahko zgodi med in vključno z 0,5 sekunde in pa do vključno 2,5 sekunde.
- Med kodo se skrivajo tudi vrstice, ki so uporabljene za »kozmetične dodatke« v igri, zato jih nisva opisala preveč podrobno.



Slika 16: Izgled "Spawnerja" v Unityju

- Tako izgleda najin »Spawner« v Unityju. Zgoraj pod »Transform« se vidi njegova začetna lokacija. Pod tem so elementi, ki se pojavijo vedno, ko se sadje ustvari (dim in pokajoči zvok). Pod tem pa sva določila v igri vse vrste sadja (jabolka, banane, češnje, jagode in limone). Nato sledijo elementi Smerx, Smery in Smerz, ki določijo smer, v kateri bo sadje letelo.

7.1.2. KODA SWORDCUTTER

```
Unity Script (1 asset reference) | 0 references
public class SwordCutterLeft : MonoBehaviour
{
    public AudioSource slice2;
    public Counter counter;
    Unity Message | 0 references
    void Start()
    {
        slice2 = GetComponent<AudioSource>();
    }
    Unity Message | 0 references
    void OnCollisionEnter(Collision collision)
    {

        slice2.Play();
        counter.Hit();

        InputDevice device = InputDevices.GetDeviceAtXRNode(XRNode.LeftHand);
        HapticCapabilities capabilities;
        if (device.TryGetHapticCapabilities(out capabilities))
            if (capabilities.supportsImpulse)
                device.SendHapticImpulse(0, 1f, 1.0f);
    }
}
```

Slika 17: Celotna koda SwordCutter

- Ta koda nam omogoči, da kadar uporabnik zadene sadje, to izgine. Prav tako doda Counter oziroma števec točk. Koda na sliki se imenuje SwordCutterLeft, kar pomeni, da je napisana za levi kontroler. Enaka koda je napisana tudi za desni kontroler, vendar se imenuje SwordCutterRight.
- Spet sva dodala AudioSource, element, ki omogoča dodajanje zvoka. Dodala sva tudi element za števec, ki se imenuje Counter.
- "Slice2.Play" predvaja zvok vedno, ko je sadje zadeto, counter.hit pa je funkcija, ki prišteje točke na števec. Koda spodaj je namenjena temu, da kontrolorja zavibrirata vedno, kadar uporabnik z njima zadene sadje.

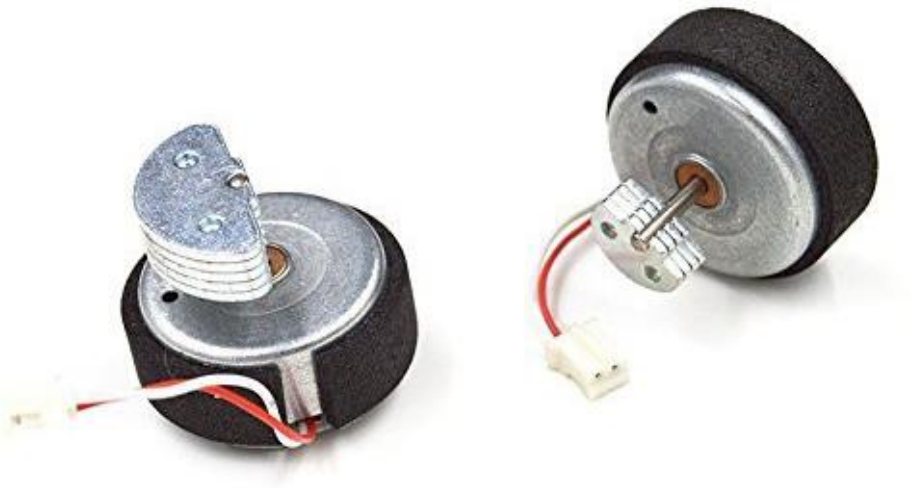
7.1.3. KODA COUNT

```
Unity Script (1 asset reference) | 2 references
public class Count : MonoBehaviour
{

    public AudioSource go;
    2 references
    public void WaitTimer(int tg)
    {
        transform.GetComponent<TextMeshPro>().enabled = true;
        transform.GetComponent<TextMeshPro>().SetText((tg-1).ToString());
        GetComponent<HapticSource>().Play();
        if (tg == 1)
        {
            transform.GetComponent<TextMeshPro>().SetText(("Go!").ToString());
            StartCoroutine(Invis());
            go.Play();

        }

    }
    1 reference
    IEnumerator Invis()
    {
        yield return new WaitForSeconds(1);
        transform.GetComponent<TextMeshPro>().enabled = false;
    }
```

Slika 18: Celotna koda Count

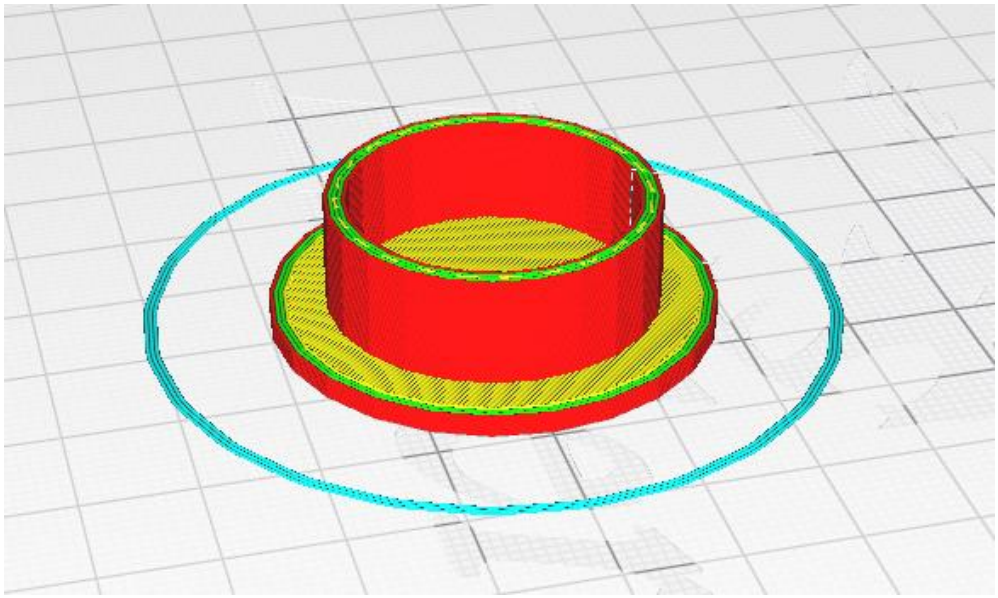
- Prva vrstica prikaže tekst za števec na ročki uporabnika, da ve, kdaj se mora v igri obrniti.
- Druga vrstica prikaže odštevanje »3, 2, 1, Go!«, da ga opozori, ko poteče 15 sekund in se mora obrniti za 180 stopinj.
- »StartCoroutine(Invis());« naredi po odštevanju tekst na ročki neviden, da uporabnika med igro ne moti.

8. IZDELAVA JOPIČA



Slika 19: Uporabljeni motorčki (viri slik: vir [8])

Uporabila sva motorčke, ki se velikokrat nahajajo v ročkah za igralne konzole in vibrirajo. Vzela sva jih iz starega kontrolorja *PlayStation 3*. Motorčki vibrirajo na takšen način, da se z veliko hitrostjo vrtijo. Operirajo z relativno nizko napetostjo (3–12 V). Povezala sva jih z Raspberry Pijem, dvema relejema in dvema 9-V baterijama. Vedno ko se sproži Raspberry Pi, se prižge rele, ki pošlje 9 V na motorčke, ki se nato prižgejo.



Slika 20: Model za motorčke

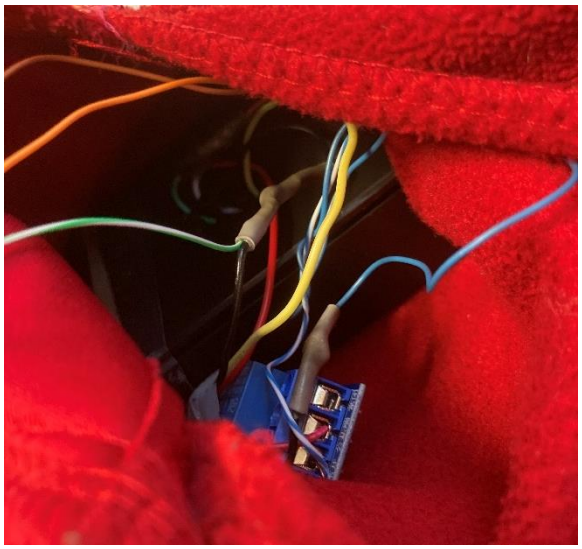
Za jopič sva uporabila telovnik, na katerega sva kasneje pritrdila vse potrebno. V okolju SketchUp sva naredila 3D-model ohišja za motorčke, ki omogočajo lažjo aplikacijo na jopič.

Slika 21: Slika zunanjosti jopiča

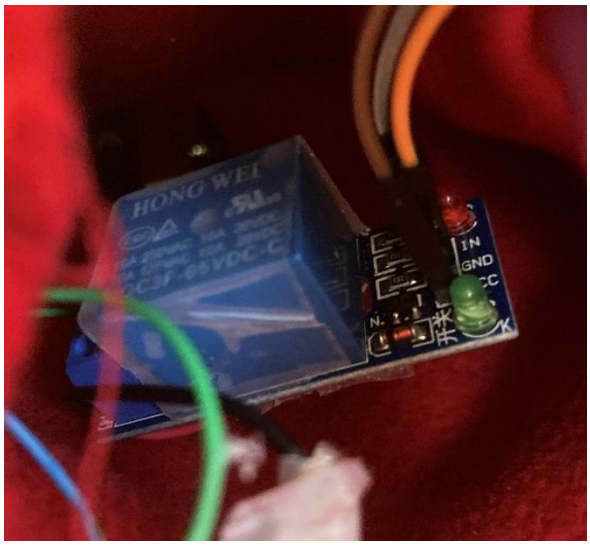
Motorčke, povezane na Raspberry Pi, releje in 9-V baterije sva nato prilepila na najin telovnik. Telovnik se napaja s pomočjo prenosne baterije firme HTC, ki jo lahko uporabnik zatakne za pas, da mu ni v napoto. Baterija omogoča napajanje na 15 W, kapaciteta baterije pa je 4700 mAh. To omogoča dolgotrajno uporabo telovnika (približno 6 ur).

Slika 22: Slika notranjosti jopiča

Za ta prototip sva se omejila le na dva motorčka, saj bi jih več zahtevalo večje napajanje. Projekt ima veliko potenciala za nadgradnjo. Motorčke sva postavila na zgornji del prsi, saj je tam telovnik najbolj oprijet, kar omogoča boljšo učinkovitost vibracij. Prilepila sva jih s pomočjo vročega lepila, ker je bil to najlažji in najboljši način pritrditve. Zraven motorčkov sva v telovnik naredila majhno luknjico za kabla, ki prihajata izven motorčkov. Za povezavo med Raspberry Pijem, releji, motorčki in baterijami sva uporabila star internetni kabel, ker so v teh kablih štirje pari enožilnih žic, ki dajejo boljšo vzdržljivost kot pa večžilna žica. Skupaj sva jih združila samo z lepilnim trakom. Po prvih poskusih sva videla, da to ne bo zadostovalo, zato sva jih najprej spajkala s cinom in jih s termokrčljivimi cevkami izolirala. Za dodatno strukturo sva še te žice ob rob telovnika pritrdila z vročim lepilom, da so skoraj neopazni in da se lahko telovnik lažje nadene.



Slika 23: Slika desnega žepa



Slika 24: Slika levega žepa

Ker ima ta telovnik žepe, sva tudi Raspberry Pi, releje in baterije skrila v notranjo stran telovnika.

8.1. POVEZAVA Z RASPBERRY PIJEM

BHaptics na spletni strani javno objavlja SDK-je, preko katerih sva lahko izvedela, na kateri IP pošilja njihov program podatke na jopiče. Z Websocketom lahko te podatke beremo. Midva sva Websocket napisala v programu Python. Za lažjo izdelavo jopiča sva v kodo dodala še to, da podatke pošilja naprej na Raspberry Pi.

8.2. KODA ZA WEBSOCKET

```python
import asyncio
import websockets
from websocket import create_connection
import json
import socket
import atexit

s = socket.socket()
ws = create_connection("ws://localhost:15881/v2/feedbacks")

def povezava(s):
    host = '172.20.10.9'
    port = 42423
    s.connect((host, port))
povezava(s)

async def server(websocket, path):
    while True:
        try:
            data = await websocket.recv()
            data = json.loads(data.replace("'", "\""))
            if 'Submit' in data:
                status = 'a'
            print(status)
            s.send(status.encode())
        except Exception as e:
            print(e)
start_server = websockets.serve(server, "localhost", 15881)
asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
def exit_handler():
    atexit.register(s.close)
```

Slika 25: Celotna koda Websocket

- Prva vrstica definira “s”, da je socket. Druga vrstica se poveže na IP naslov, na katerem se pošiljajo podatki iz že prej omenjenega »Bhaptics Playerja«. Funkcija “povezava” ustvari povezavo z Raspberry pijem. Na koncu skliče na funkcijo “povezava”.
- “async def” določi funkcijo. Nato sva napisala neskončno zanko, da v neskončnost bere podatke iz Websocketa. Nato prevede “json” v jezik, ki ga lahko program Python prebere in pogleda, če je v podatkih “Submit”. Če je, v status zapiše “a”. Če podatkov ne zazna, izpiše napako.

8.3. KODA ZA RASPBERRY PI

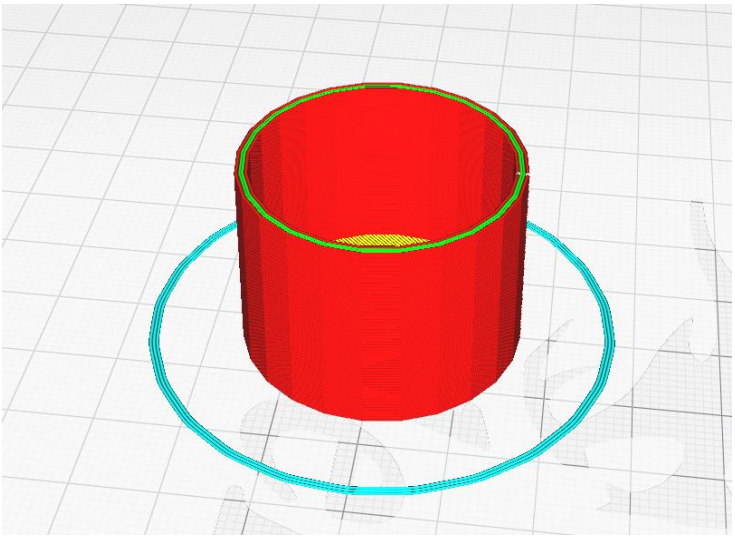
```
import socket
import RPi.GPIO as GPIO
import time
import atexit

GPIO.setmode(GPIO.BCM)
GPIO.setup(22, GPIO.OUT)

def Main():
    host = '172.20.10.9'
    port = 42424
    s = socket.socket()
    s.bind((host, port))
    s.listen(1)
    c, addr = s.accept()
    while True:
        data = c.recv(1024).decode()
        print(data)
        if 'a' in data:
            GPIO.output(22, GPIO.HIGH)
            time.sleep(0.4)
            GPIO.output(22, GPIO.LOW)
        if not data:
            break
        data = str(data).upper()
    c.close()
if __name__ == '__main__':
    Main()
def exit_handler():
    atexit.register(c.close)
```

Slika 26: Celotna koda Raspberry Pi

- S “host” se najprej računalnik poveže na IP od Raspberry Pija in mu tako pošilja podatke iz najine igre. Raspberry Pi dobi podatke, in če so ustrezni, preko knjižnice RPi.GPIO prižge pin, ki potem prižge rele.

9. UGOTOVITVE IN IZBOLJŠAVE

Med testiranjem sva ugotovila možnost izboljšave, in sicer pri motorčkih. Ker motorčki povzročajo vibracije s tem, da se vrtijo, so s seboj povlekli lase ene izmed testirank. Na srečo se ni zgodilo nič resnega, vendar nama je to vseeno dalo idejo za izboljšavo telovnika. Takoj za tem sva 3D zmodelirala pokrovčka, ki gresta čez motorčka, tako da sta nevidna in nedosegljiva.

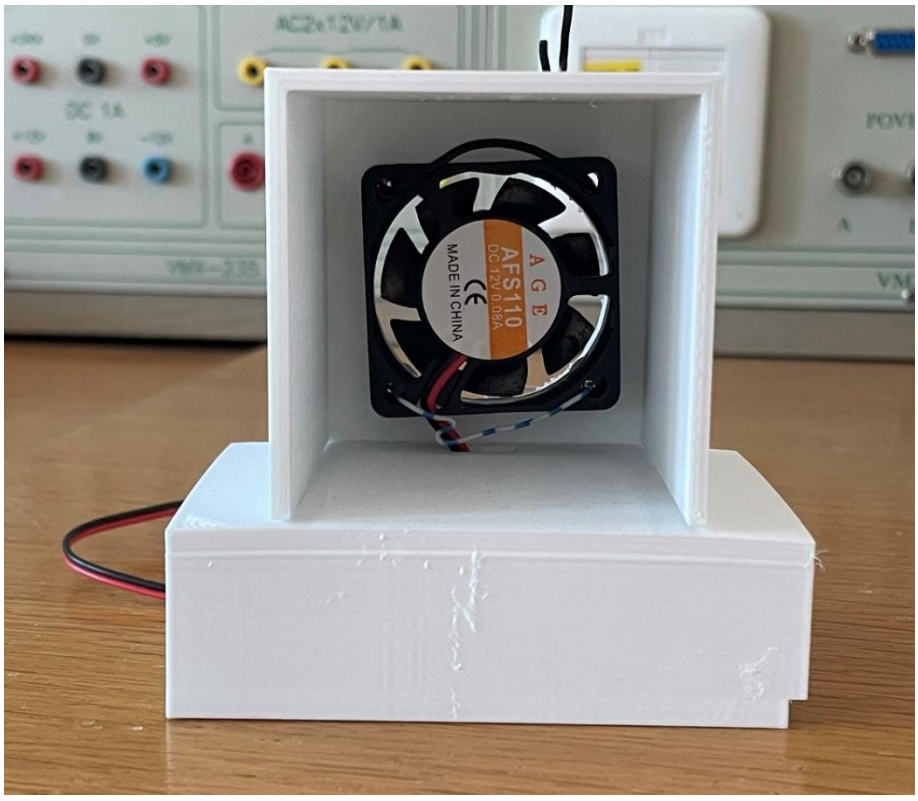


Slika 27: Slika pokrovčka motorčkov

Slika 28:Slika pokrovčkov na telovniku

10. IZDELAVA SMELL-O-VISIONA

Za prvi prototip vonjav sva ustvarila 3D-model, ki ga sestavlja ohišje za Arduino, in prostor, v katerega sva postavila dišavo. Ta model po prvem poskusu ni bil najboljši, saj sva se malo zmotila pri merah.

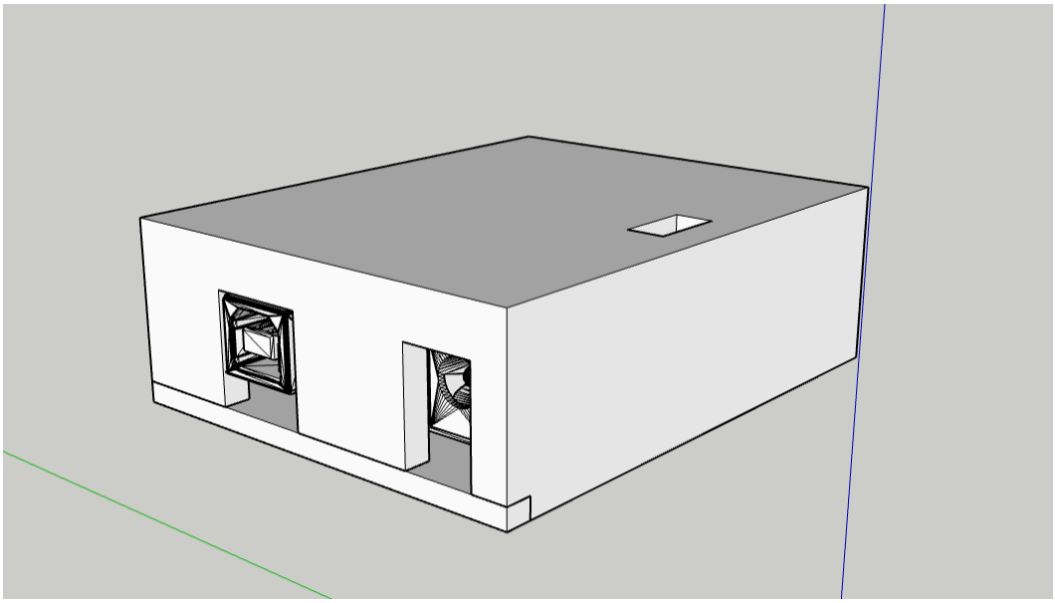


Slika 29: Slika prototipa Smell-O-Visiona

Da v najini igri simulira vonj sadja, sva uporabila eterično olje z vonjem višnje Za posodico z dišavo je 40-mm ventilator, ki se sproži vedno, ko mu Arduino pošlje signal, torej vedno ko je sadje zadeto. Arduino je povezan na računalnik z USB-kablom, preko katerega dobi napajanje in podatke.

Slika 30: 40-mm ventilator (viri slik: vir [9])

Po izdelavi prototipa sva v Sketch-upu ponovno naredila ohišje Arduino, tokrat s pravimi merami.



Slika 31: Popravljeno ohišje Arduino v Sketch-upu

Naredila sva tudi novo ohišje za ventilatorje. Tokrat sva za boljši prenos vonja do uporabnika uporabila 120-mm ventilator in dva 40-mm ventilatorja. Obe velikosti ohišij sva zmodelirala v Sketch-upu in ju nato 3D natisnila.
Kupila sva tudi več različnih eteričnih olj, in sicer z vonjem višnje, limonske trave, brinovih jagod in vrtnice z jagodičevjem. Uporabila sva več vonjav hkrati, da je bil učinek močnejši.

Slika 32: Ohišje 120- in 40-mm ventilatorja

Ugotovila sva, da je vonj limonske trave najmočnejši izmed vseh vonjav, zato sva ga postavila v ohišje s 40-mm ventilatorjem, saj ne potrebuje tako močnega ventilatorja. Šibkejše dišave sva kombinirano postavila v ohišje s 120-mm ventilatorjem, da je bil njihov vonj intenzivnejši.
Ugotovila sva, da dišave postanejo intenzivnejše, če jih mešamo z vodo, zato sva to storila; tudi eteričnega olja sva tako porabila manj. Voda je izpodrinila olje na površje, zaradi česar je bilo bližje ventilatorjem. To pomeni, da je ventilator dišavo do uporabnika hitreje prenesel.

En ventilator sva postavila pred igralca, drugega pa za njega. Igra je namreč narejena tako, da se po petnajstih sekundah smer streljanja sadja iz topov spremeni, zato se igralec obrača. Zaradi tega je tudi zelo pomembno, da je telovnik brezžičen in da deluje na powerbank. Ko se uporabnik obrne nazaj, se vklopi ventilator, ki je za njim, in pa obratno. Ta dinamika v igri tudi pomaga, da vonj ni preveč močan in se različni ne mešajo preveč med seboj, saj iz ene strani prihajajo drugačne vonjave kot iz druge. Vonjave potrebujejo nekaj časa, da igralca dosežejo, zato med igro postanejo vedno bolj intenzivne.

10.1. POVEZAVA ARDUINO IN KODA COUNTER

```
Unity Script (1 asset reference) | 3 references
public class Counter : MonoBehaviour
{
    public int counter = 0;
    public string portName = "COM3";
    public CounterVzadi CounterVzadi;
    SerialPort arduino;

    Unity Message | 0 references
    void Start()
    {

        arduino = new SerialPort(portName, 9600);
        arduino.Open();
        arduino.Write("a");

    }
    2 references
    public void Hit()
    {
        counter++;
        transform.GetComponent<TextMeshPro>().SetText((counter).ToString());
        CounterVzadi.UpdateVzadi(counter);
    }
    Unity Message | 0 references
    private void OnApplicationQuit()
    {
        arduino.Write("c");
        Debug.Log(counter.ToString());
    }
}
```

Slika 33: Celotna povezava Arduino in koda Counter

- Del kode void Start() definira, na kateri USB-priključek pošilja podatke. Število 9600 je velikost paketa, ki ga pošlje na Arduino. Arduino po pridobljenih podatkih določi, ali bo ventilator na najini različici Smell-O-Visiona prižgan ali izklopljen – ali bo aroma močna ali ne.
- Funkcija »public void Hit()« doda eno točko, vedno ko uporabnik zadane sadje. Nato spremeni števec, ki je pred uporabnikom in za njim tako, da mu pokaže število doseženih točk.
- Zadnja funkcija »private void OnApplicationQuit()« pošlje Arduinu signal, da Arduino ugasne ventilatorja, »Debug.Log(counter.ToString());« pa v konzoli izpiše število doseženih točk.

10.2. KODA ZA ARDUINO

```
const int Vent1= 13;
char zacni = 'c';

void setup() {
  pinMode(Vent1, OUTPUT);
  Serial.begin(9600);
  digitalWrite(Vent1, HIGH);

}


  void loop()
  {
    if (Serial.available() > 0)
    {
    zacni = Serial.read();
    while (zacni == 'a')
    {
      digitalWrite(Vent1, LOW);
      delay(19000);
      digitalWrite(Vent1, HIGH);
      delay(19000);
      zacni = Serial.read();
     }
    if (zacni == 'c')
   {
      digitalWrite(Vent1, HIGH);
   }
    }
  }
```

Slika 34: Koda za Arduino

- Koda pridobi podatek o stanju v igri preko USB-povezave in potem določi status ventilatorja (prižgan ali ugasnjen). Za prižig potrebujeva v bistvu samo en 'pin output' zaradi vrste releja, ki sva ga uporabila. Ta rele samo določa, na katero stran gre napetost; s tem sva lahko uporabila dva ventilatorja na en pin in en rele. Z »Vent1 = 13;« sva določila, na kateri pin pošilja signal za prižig ventilatorja. Na vsakih 19 sekund se zamenja, kateri ventilator je prižgan.

11. OPIS TESTIRANJA

Testiranje sva začela s tem, da sva 20 subjektov razdelila na 4 skupine po 5. Skupine so se imenovale A, B, C in D. Pri vsaki skupini sva si zapisovala rezultate posameznikov, da sva dobila podatke o tem, v kakšnih razmerah so bili subjekti pri igranju na splošno bolj zbrani oz. natančni.

Igre sva razdelila na štiri kategorije, in sicer:

- samo VR-gra
- VR-igra in vonjave
- VR-igra in jopič
- VR-igra z vonjavami in jopičem

Uporabila sva strategijo, ki se po angleško imenuje 'randomized crossover design'. To sva storila zato, da bi rezultati bili čim bolj objektivni, saj s to strategijo skupine začnejo z različnimi razmerami igranja, s čimer pridobimo bolj naključne rezultate, ki si ne sledijo po točnem zaporedju. To sva storila tudi zato, ker sva se zavedala, da bodo igralci tekom testiranja v igranju postali bolj vešči, kar pomeni npr., da če bi vsi začeli najprej samo z VR igro, bi proti koncu skoraj vsi dosegli višje število točk pri VR-igri z vonjavami in jopičem kot pri prvi igri.

Igralci so imeli na voljo eno minute za vsako kategorijo, v kateri so poskušali doseči čim višji rezultat. Pred igro sva vsem na enak način razložila navodila igre, nato pa jih pri igranju nisva več motila. To nama je dalo podatek o tem, kako uspešno se bodo igralci spomnili, da se morajo po 15 sekundah obrniti – ali jim bodo vibracije na jopiču pri tem bolj pomagale, kot pa če jih ne bi bilo.

KAKO BI TESTIRANCI OCENILI POSAMEZNO IZKUŠNJO

Na koncu sva subjektom zastavila še hipotetično vprašanje. Rekla sva jim, da so se znašli v arkadi, kjer lahko igrajo najino igro. Na voljo imajo 20 žetonov. Samo VR-igra brez vsega stane 1 žeton, VR-igra z vonjavami ali VR-igra z jopičem stane 2 žetona, VR-igra z vsem skupaj pa stane 5 žetonov. Zanimalo naju je, kako bodo posamezniki žetone razdelili. S tem sva pridobila podatek, kateri del najinega projekta jim je bil najbolj všeč, glede na znesek in število žetonov.

12. REZULTATI TESTIRANJA

V prvi celici so točke, ki so jih uporabniki dosegli samo z igranjem brez dodatnih faktorjev. Druga celica predstavlja poskus, pri katerem so testiranci igrali samo s pomočjo vonjav. Tretja celica predstavlja poskus, pri katerem so imeli oblečen samo jopič, četrta celica pa ponazarja točke, ki so jih pridobili s pomočjo vseh faktorjev skupaj (vonjavami ter jopičem).

Skupina A

	1 – samo VR-igra	2 – VR-igra z vonjavami	3 – VR-igra z jopičem	4 – VR-igra z vonjavami in jopičem
		A		
Uporabnik 1 A	**59**	**40**	**50**	**56**
Uporabnik 2 A	**49**	**52**	**65**	**76**
Uporabnik 3 A	**57**	**50**	**74**	**80**
Uporabnik 4 A	**65**	**86**	**78**	**80**
Uporabnik 5 A	**57**	**64**	**67**	**66**

Tabela 1: Rezultati skupine A

Skupina B

		B		
Uporabnik 1 B	**75**	**76**	**73**	**86**
Uporabnik 2 B	**79**	**63**	**70**	**77**
Uporabnik 3 B	**49**	**62**	**54**	**65**
Uporabnik 4 B	**71**	**64**	**75**	**78**
Uporabnik 5 B	**87**	**46**	**86**	**78**

Tabela 2: Rezultati skupine B

Skupina C

		C		
Uporabnik 1 C	**92**	**95**	**80**	**88**
Uporabnik 2 C	**82**	**76**	**65**	**67**
Uporabnik 3 C	**88**	**90**	**64**	**70**
Uporabnik 4 C	**87**	**82**	**53**	**90**
Uporabnik 5 C	**68**	**81**	**53**	**68**

Tabela 3: Rezultati skupine C

Skupina D

D				
Uporabnik 1 D	**69**	**78**	**75**	**72**
Uporabnik 2 D	**62**	**60**	**66**	**64**
Uporabnik 3 D	**75**	**82**	**80**	**78**
Uporabnik 4 D	**48**	**58**	**72**	**73**
Uporabnik 5 D	**72**	**77**	**80**	**74**

Tabela 4: Rezultati skupine D

Ko sva pridobila vse rezultate, sva jih med sabo primerjala v Excelu s posebno enačbo, ki se imenuje T-test:

- =T.TEST(matrika1; matrika2; rep; vrsta)

S tem sva lahko primerjala željene rezultate testiranja med seboj, z uporabo p-vrednosti ali stopnje značilnosti.

12.1. KAJ JE P-VREDNOST ALI STOPNJA ZNAČILNOSTI?

P-vrednost (stopnjo značilnosti) uporabljamo pri preverjanju hipotez. V veliki večini primerov pri kvantitativnih raziskavah ne moremo anketirati vseh enot populacije. Zaradi tega običajno iz populacije izberemo reprezentativni vzorec enot oziroma testirancev, na katerih opravimo raziskavo, kjer pridobimo podatke. Na podlagi testirancev iz vzorca želimo preverjati hipoteze na nivoju populacije. Pri tem nam pomaga stopnja značilnosti ali p-vrednost. [9]

12.2. KAJ NAM POVE?

Najpogosteje uporabljamo stopnjo značilnosti v višini 0,05. Če rezultati kažejo statistično značilno pomembne razlike pri stopnji značilnosti v višini 0,05, lahko zaupamo in verjamemo, da statistično značilne razlike res obstajajo. Če je p-vrednost manjša ali enaka 0,05, potem lahko z veliko gotovostjo (95 %) posplošimo rezultate iz vzorca na populacijo. [9]

12.3. KAJ ČE JE P-VREDNOST VEČJA OD 0,05?

Če je p-vrednost večja od 0,05, se moramo vzdržati vsakršnega sklepanja iz vzorca na populacijo in moramo rezultate interpretirati le na nivoju vzorca. V primeru p>0,05 torej določenega rezultata na nivoju vzorca ne moremo posplošiti na nivo populacije. [9]

12.4. NAJINE UGOTOVITVE NA PODLAGI P-VREDNOSTI

Najprej sva primerjala dosežene točke samo VR z doseženimi točkami VR igre z vonjavami. Nato sva primerjala dosežene točke samo VR igre z doseženimi točkami VR igre z jopičem. Na koncu sva primerjala še dosežene točke samo VR igre in dosežene točke VR igre z vonjavami in jopičem. Dobila sva naslednji rezultat. [9]

VR-igra z vonjavami	VR-igra z jopičem	VR-igra z vonjavami in jopičem
0.44	0.43	0.05

Tabela 5: P-vrednosti

Rezultati so pokazali, da VR-igre z vonjavami in VR-igre z jopičem ne moremo posplošiti na nivoju populacije, saj je p-vrednost večja od 0.05, medtem ko je VR-igra z vonjavami in jopičem dosegla p-vrednost natanko 0.05, kar nam pove, da lahko te rezultate na vzorcu populacije posplošimo. S tem sva dokazala, da je VR-igra z vonjavami in jopičem testirancem res pomagala pridobiti boljše rezultate. Pri VR-igri z vonjavami in VR-igri z jopičem bi pa za to ugotovitev morali testirati več ljudi.

12.5. KAKO BI TESTIRANCI OCENILI POSAMEZNO IZKUŠNJO?

Kot že prej opisano, sva nekaj izmed testirancev vprašala, kako bi razdelili svojih 20 žetonov na najini igri, če bi se znašli v arkadi. Dobila sva naslednje ugotovitve.

	Samo VR-igra	VR-igra z vonjavami	VR-igra z jopičem	VR-igra z vonjavami in jopičem
Uporabnik 1				4 x 5 žetonov
Uporabnik 2			10 x 2 žetona	
Uporabnik 3	20 x 1 žeton			
Uporabnik 4		10 x 2 žetona		
Uporabnik 5				4 x 5 žetonov
Uporabnik 6		2 x 2 žetona	3 x 2 žetona	2 x 5 žetonov
Uporabnik 7				4 x 5 žetonov
Uporabnik 8		5 x 2 žetona		2 x 5 žetonov
Uporabnik 9				4 x 5 žetonov
Uporabnik 10	20 x 1 žeton			
Uporabnik 11				4 x 5 žetonov

Tabela 6: Porazdelitev žetonov

Kot lahko razberemo iz tabele, so odgovori raznoliki. Testiranci so vrednotili posamezne izkušnje različno glede na svoje preference.

Zbrano število žetonov pri posameznih kategorijah:

- Samo VR-igra je zbrala skupaj 40 žetonov;
- VR-igra z vonjavami je zbrala 34 žetonov;

- VR-igra z jopičem je zbrala 26 žetonov;
- VR-igra z vonjavami in jopičem je zbrala 120 žetonov.

Kot lahko vidimo iz zbranega števila žetonov, je na prvem mestu z veliko prednostjo VR-igra z vonjavami in jopičem. Kljub temu, da s takšno porazdelitvijo žetonov igrajo manj iger, se je večina odločila za to izkušnjo. Drugo mesto zbranih žetonov je dobila samo VR-igra, in sicer 40 zbranih žetonov. Oba izmed testirancev sta po vprašanju, zakaj sta se tako odločila, odgovorila, da sta želela imeti v igri čim več poskusov, saj je bila ta kategorija najcenejša, in sicer 1 žeton, kar jima je dalo 20 poskusov. Naslednja kategorija je bila VR-igra z vonjavami, ki je zbrala 34 žetonov, za njo pa VR-igra z jopičem s 26 žetoni.

Legenda:
4 – VR-igra z jopičem in vonjavami
3 – VR-igra z jopičem
2 – VR-igra z vonjavami
1 – samo VR-igra

Grafikon 1: Število zbranih žetonov pri posameznih kategorijah

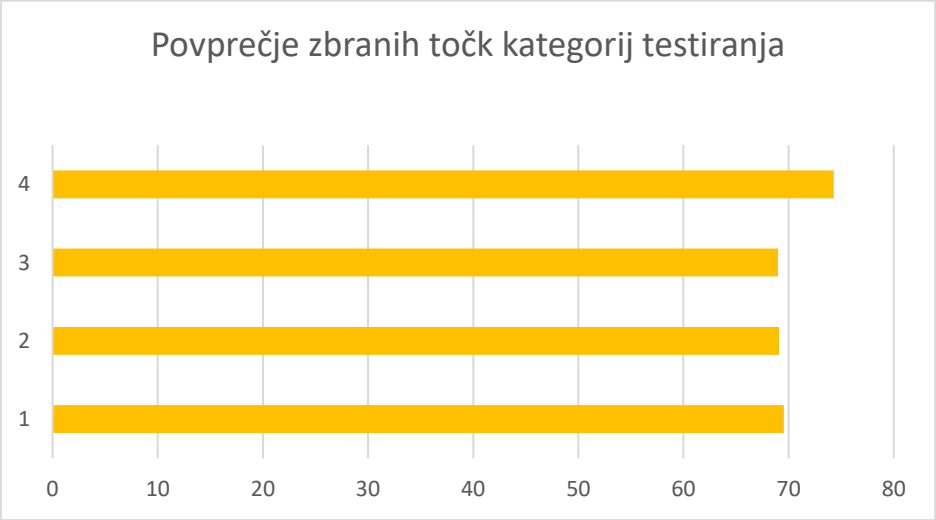
13. OBRAZLOŽITEV HIPOTEZ

13.1. PRVA HIPOTEZA

Vonjave in haptični jopič bodo imele na uspešnost igralca v igri pozitiven učinek.

Povprečje vseh točk, ki so jih dosegli testiranci pri posameznih poizkusih, je pri uporabi jopiča ter vonjav bistveno višje. Iz tega lahko sklepamo, da so vonjave in jopič res pomagale testirancem pri igri in da hipoteza drži.

Legenda:
4 – VR-igra z jopičem in vonjavami
3 – VR-igra z jopičem
2 – VR-igra z vonjavami
1 – Samo VR-igra



Grafikon 2: Povprečje zbranih točk kategorij testiranja

Kot lahko razberemo iz grafikona, je igra z jopičem ter vonjavami imela v povprečju zbranih največ točk. Povprečje VR igre z vonjavami in jopičem je znašalo 74,3 točk, medtem ko so bila povprečja naslednjih kategorij sledeča: VR-igra z jopičem je zbrala 69 povprečnih točk, VR-igra z vonjavami je zbrala 69,1 povprečnih točk, in samo VR-igra je zbrala 69,55 povprečnih točk.

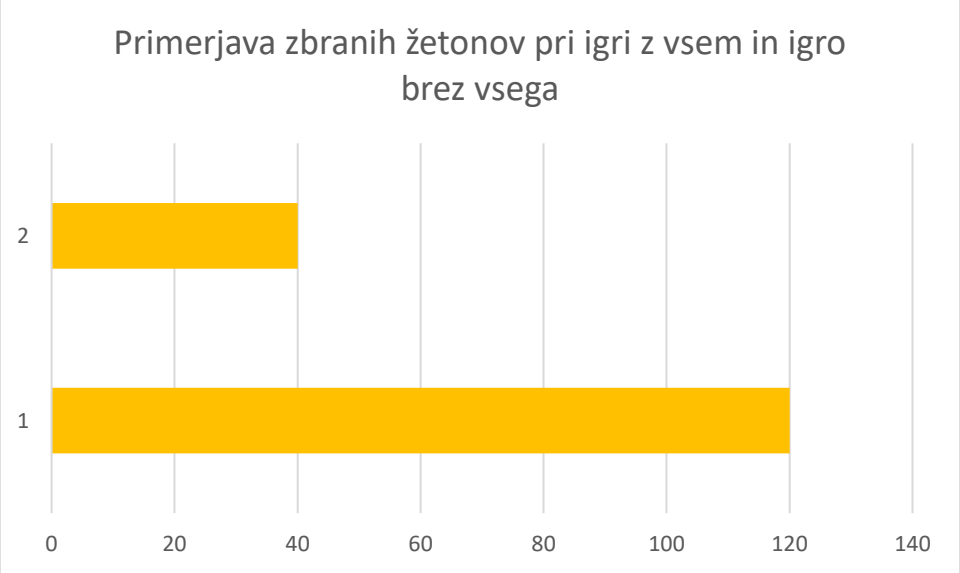
13.2. DRUGA HIPOTEZA

Vonjave in haptični jopič bodo imele na subjektivno izkušnjo igralca pozitiven učinek.

To hipotezo sva lahko preverila s porazdelitvijo žetonov. Dobila sva naslednje rezultate.

Legenda:

2 – samo VR-igra
1 – VR-igra z vonjavami in jopičem



Grafikon 3: Primerjava zbranih žetonov pri igri z vsem in igro brez vsega

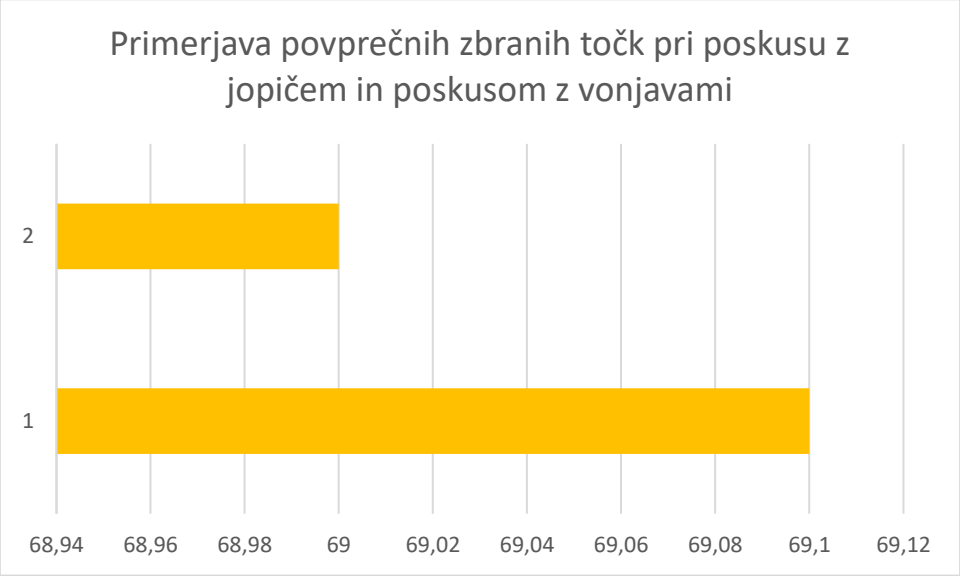
Kot lahko vidimo, je samo VR-igra zbrala 40 žetonov, med tem ko je VR-igra z vonjavami in jopičem zbrala 120 žetonov. S tem lahko potrdiva svojo hipotezo, da je izkušnja z jopičem in vonjavami uporabnikom pustila bolj pozitiven učinek kot pa izkušnja brez.

13.3. TRETJA HIPOTEZA

Učinek vonjav bo imel bolj pozitiven učinek na uspešnost igralca v igri kot haptični jopič.

Odgovori sošolcev, ki so testirali igro najprej brez vonjav in potem z njimi, so bili skoraj enotni, da vonjav niso opazili, dokler jim jih nisva omenila. Večina jih je bila preveč zavzeta z igranjem, da bi jih opazili. Eden izmed faktorjev, zakaj jih niso, je verjetno tudi ta, da so vonjave potrebovale nekaj časa, da so do njih pripotovale, saj so bili ventilatorji kar oddaljeni, vonjave pa same po sebi tudi niso bile zelo močne, vendar je povprečno doseženo število točk pokazalo drugače.

Legenda:
2 – VR-igra z jopičem
1 – VR-igra z vonjavami



Grafikon 4: Primerjava povprečnih zbranih točk pri poskusu z jopičem in poskusom z vonjavami

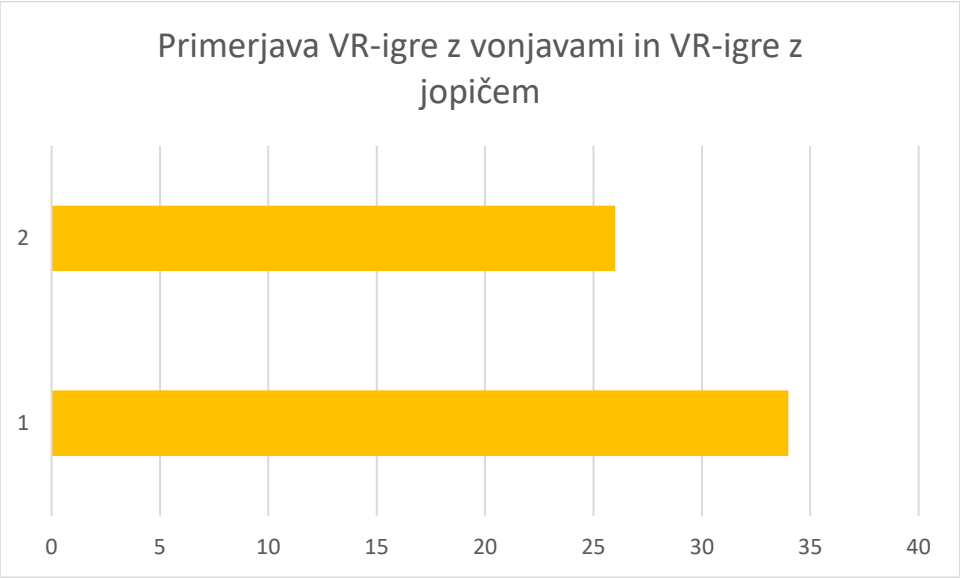
Po seštevku rezultatov sva lahko razbrala, da je poskus z vonjavami zbral za 0,1 več povprečnih zbranih točk kot poskus z jopičem. Razlika ni velika, vendar vseeno dokaže, da najina hipoteza drži. Očitno so vonjave igralcem le pomagale – morda so jih opazili podzavestno.

13.4. ČETRTA HIPOTEZA

Izkušnja z vonjavami bo imela bolj pozitiven učinek na subjektivno izkušnjo igralca kot izkušnja s haptičnim jopičem.

Ta podatek sva zopet lahko razbrala iz testirančevih razporeditev žetonov. Ugotovila sva, da je VR-igra z vonjavo dosegla skupno 34 žetonov, VR-igra z jopičem pa 26. To pomeni, da hipoteza drži, saj so igralci kljub isti ceni obeh izkušenj svoje žetone raje porabili za izkušnjo z vonjavami.

Legenda:
2 – VR-igra z jopičem
1 – VR-igra z vonjavami



Grafikon 5: Primerjava VR igre z vonjavami in VR-igre z jopičem

14. ZAHVALA

Za pomoč pri raziskovalni nalogi bil se zahvalila:

- mentorjema Samu Železniku in Roku Urbancu za pomoč pri izdelavi in celotni izvedbi projekta;
- profesorju Roku Urbancu pri pomoči pri snemanju videoposnetka uporabe;
- profesorici Sonji Lubej za lektoriranje slovenskega besedila;
- profesorici Beti Tomic za lektoriranje angleškega besedila;
- dijakom, ki so nama pomagali pri testiranju.

15. ZAKLJUČEK

Virtualna resničnost je zelo zanimiva, saj uporabnika v svet igre zelo pritegne. S to raziskovalno nalogo sva dokazala, kako s pomočjo fizičnih dodatkov za stimulacijo tipa in čuta naredimo ta virtualni svet še bolj zanimiv in realističen. Pri izdelavi raziskovalne naloge sva se naučila kar nekaj novih veščin, kot npr. povezavo med računalniki, upravljanje Arduina in Raspberry Pija, ter izdelave iger v Unityju za VR. Projekt ima veliko možnosti za nadgradnjo, katere bi v prihodnosti rada raziskala.

16. POVZETEK

V današnjem svetu sta zabava in sprostitev vedno bolj pomembni in vedno znova iščemo izhode iz našega vsakdanjega življenja. Nekateri pobegnejo v virtualno resničnost, zato je bil najin cilj v tej raziskovalni nalogi odkriti, kako bi bila uporaba virtualne resničnosti bolj realistična in poglobljena. To sva storila s prikazom haptične odzivnosti telesa na tresljaje in vonjave. Naredila sva igro, ki omogoča zabavno in bolj poglobljeno VR-doživetje. Haptično odzivnost sva dosegla z motorčki, pritrjenimi na jopič uporabnika, ki povzročajo tresljaje in mu tako omogočajo, da čuti elemente igre na telesu fizično. Dodala sva tudi izkušnjo z vonjavami, pri kateri uporabnik s pomočjo dejanske arome eteričnega olja v prostoru podoživlja vonj v navidezni resničnosti igre. Igro tako zaznava s čuti in to njegovo doživljanje sicer fiktivne realnosti postane intenzivnejše in s tem bolj sproščujoče.

17. VIRI

17.1. VSI UPORABLJENI ASSETI

1. https://assetstore.unity.com/packages/3d/vegetation/trees/low-poly-tree-pack-57866
2. https://assetstore.unity.com/packages/3d/vegetation/trees/free-trees-103208
3. https://assetstore.unity.com/packages/3d/environments/landscapes/simple-low-poly-nature-pack-157552
4. https://assetstore.unity.com/packages/3d/props/weapons/stylish-cannon-pack-174145
5. https://assetstore.unity.com/packages/vfx/shaders/free-skybox-extended-shader-107400
6. https://assetstore.unity.com/packages/3d/environments/landscapes/low-poly-simple-nature-pack-162153
7. https://assetstore.unity.com/packages/3d/props/food/low-poly-fruit-pickups-98135
8. https://assetstore.unity.com/packages/3d/props/free-beach-essentials-asset-pack-131149
9. https://assetstore.unity.com/packages/3d/props/exterior/super-beach-pack-39084
10. https://assetstore.unity.com/packages/tools/integration/bhaptics-haptic-plugin-76647
11. https://assetstore.unity.com/packages/tools/particles-effects/melee-weapon-trail-1728
12. https://assetstore.unity.com/packages/3d/props/weapons/watermelon-sword-191078
13. https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647

17.2. VIRI LITERATURE:

1. Vir[1]: Determinants of Diffusion of Virtual Reality, Namron Regrebsubla, 2016
(28. 2. 2022)
2. Vir[2]: Pimentel, K., & Teixeira, K. (1993). Virtual reality. New York, NY: McGraw-Hill.
(28. 2. 2022)
3. Vir[3]: Sutherland, I. E. (1968) "A head-mounted three dimensional display" - https://web.archive.org/web/20180914060812/https://cacs.usc.edu/education/cs653/Sutherland-HeadmountedDisplay-AFIPS68.pdf
(8.3.2022)
4. Vir[4]: https://virtuality.com/core-technology/
(7.3.2022)
5. Vir[5]: https://www.ultraleap.com/company/news/blog/what-is-haptics/
(8.3.2022)
6. Vir[6]: https://segaretro.org/images/e/e2/Fonz_service_manual.pdf
(8.3.2022)
7. Vir[7]: https://onitama.tv/gamemachine/pdf/19831201p.pdf#page=17
(9.3.2022)
8. Vir[8]: https://n64squid.com/nintendo-64-patent-rumble-pak/
(10.3.2022)
9. Vir[9]: https://patents.google.com/patent/US20130215024A1/en
(10.3.2022)
10. Vir[10]: https://archive.org/details/GamePro_Issue_064_November_1994/page/n282/mode/2up
(5.3.2022)
11. Vir[11]: https://web.archive.org/web/20070323025822/http://www.forbes.com/businesswire/feeds/businesswire/2007/03/07/businesswire20070307005996r1.html
(3.3.2022)
12. Vir[12]: https://www.bhaptics.com/
(28. 2. 2022)
13. Vir[13]: A. H. WEILER (12 julij 1959). "PASSING PICTURE SCENE: Trailing 'Scent of Mystery' Through Spain -- Espionage -- Other Matters". New York Times. p. X5.
(10.3.2022)
14. Vir[14]: https://hapticsol.com/cilia/cilia
(28. 2. 2022)
15. Vir[15]: https://www.vive.com/eu/product/vive-pro/
(28. 2. 2022)
16. Vir[16]: https://www.azurefilm.si/3d-tiskalnik-creality-ender-5?gclid=CjwKCAiAgvKQBhBbEiwAaPQw3BnU-HcMFN4ohApyLgyLXWBp61iFR_c0aKAyvAygtZju9jL_dgtLxhoC7cYQAvD_BwE
(28. 2. 2022)

17. Vir[17]: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-4-Product-Brief.pdf
(28. 2. 2022)
18. Vir[18]: https://www.rs-online.com/designspark/what-is-arduino-uno-a-getting-started-guide
(28. 2. 2022)
19. Vir[19]: https://www.statistik.si/p-vrednost/
(1. 3. 2022)

17.3. VIRI SLIK:

1. Slika 1: http://blog.bigimmersive.com/wp-content/uploads/2019/06/preview.jpg
2. Slika 2: https://www.bhaptics.com/_next/static/images/img-tact-suit-x-40-003-1335b809b17b276aa0061446342e9e6b.jpg
3. Slika 3: https://images.squarespace-cdn.com/content/v1/5cad54d7b914496fea8ec210/1603602969746-9Y4RXM21BCOFFEYW8CVW/CiliaLightBlueBackground.png?format=2500w
4. Slika 4: https://www.mimovrste.com/i/58962206/700/700
5. Slika 5: https://3dshark.si/wp-content/uploads/2020/04/ender-5-creality-02.jpg
6. Slika 6: https://www.robot-advance.com/EN/ori-raspberry-pi-4-model-b-4go-2640.jpg
7. Slika 7: https://upload.wikimedia.org/wikipedia/commons/3/38/Arduino_Uno_-_R3.jpg
8. Slika 21: https://m.media-amazon.com/images/I/41cFXKiLv8L.jpg
9. Slika 28: https://image.made-in-china.com/2f0j00WHoUqatEsCbN/40mm-Fan-4007-DC-Axial-Computer-Compact-CPU-Fan-Cooler.jpg

18. PRILOGE

1. Videoposnetek uporabe: https://www.youtube.com/watch?v=BYfhW3DpRtU
2. Pri izdelavi igre sva si pomagala z dokumentacijo Unity: https://docs.unity3d.com/Manual/index.html

18.1. OPISANE KODE

1. KODA FRUITSPAWNER:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Unity.Audio;


public class FruitSpawner : MonoBehaviour
{
   public Transform smoke;
   public Transform smoke2;
   public AudioSource pop;
   public GameObject[] fruitPrefab;
   public float smerx = 5, smery = 5, smerz = 5;
   int sadje = -1;
   public Count count;
   public int spawner1 = 0;
   public FruitSpawner2 drugi;
   public Counter counter;
   public void StartGame()
   {
      StartCoroutine(Wait());
   }
   IEnumerator Wait()
   {
      for (int t1 = 4; t1 > 0; t1--)
      {
         count.WaitTimer(t1);
         yield return new WaitForSeconds(1);
      }
      if (spawner1 == 0)
      {
         StartCoroutine(SpawnFruit());

      }
      else
      {
         drugi.ZacniDrugo();
      }
   }
   public void ZacniPrvo()
   {
      StartCoroutine(SpawnFruit());
   }
```

```
  IEnumerator LetThereBeFruit()
  {
     for (int t = 0; t <15 ; t++)
     {

        yield return new WaitForSeconds(1);
     }
     spawner1 = 1;
     StartCoroutine(Wait());
  }
  public IEnumerator SpawnFruit()
  {
     int c = 0;
     spawner1 = 0;
     StartCoroutine(LetThereBeFruit());
     while (spawner1==0)
     {
        pop.Play();
        sadje = Random.Range(0, fruitPrefab.Length);
        GameObject go = Instantiate(fruitPrefab[sadje]);
        Rigidbody temp = go.GetComponent<Rigidbody>();

        if (c == 0)
        {

           smoke.GetComponent<ParticleSystem>().Play();
           c++;
        }
        else
        {

           smoke2.GetComponent<ParticleSystem>().Play();
           c = 0;
        }

        temp.velocity = new Vector3(Random.Range(smerx - 0.2f, smerx + 0.2f),
Random.Range(smery - 0.2f, smery + 0.2f), Random.Range(smerz - 0.2f, smerz + 0.2f));
        temp.angularVelocity = new Vector3(Random.Range(-5f, 5f), Random.Range(-2f, 2f),
Random.Range(-3.5f, 3.5f));
        temp.useGravity = true;

        Vector3 pos = transform.position;

        go.transform.position = pos;
        yield return new WaitForSeconds(Random.Range(0.5f, 2.5f));
     }

  }


}
```

2. KODA SWORDCUTTER

```
using UnityEngine;
using System.Collections;
using Valve.VR;
using UnityEngine.XR;
using Unity.Audio;


public class SwordCutterLeft : MonoBehaviour
{
   public AudioSource slice2;
   public Counter counter;
   void Start()
   {
      slice2 = GetComponent<AudioSource>();
   }
   void OnCollisionEnter(Collision collision)
   {

      slice2.Play();
      counter.Hit();

      InputDevice device = InputDevices.GetDeviceAtXRNode(XRNode.LeftHand);
      HapticCapabilities capabilities;
      if (device.TryGetHapticCapabilities(out capabilities))
         if (capabilities.supportsImpulse)
            device.SendHapticImpulse(0, 1f, 1.0f);
   }
}
```

3. KODA COUNT

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using Bhaptics.Tact.Unity;
using Unity.Audio;
using System.Threading;

public class Count : MonoBehaviour
{

   public AudioSource go;
   public void WaitTimer(int tg)
   {
      transform.GetComponent<TextMeshPro>().enabled = true;
      transform.GetComponent<TextMeshPro>().SetText((tg-1).ToString());
      GetComponent<HapticSource>().Play();
      if (tg == 1)
      {
         transform.GetComponent<TextMeshPro>().SetText(("Go!").ToString());
         StartCoroutine(Invis());
         go.Play();

      }

   }
   IEnumerator Invis()
   {
      yield return new WaitForSeconds(1);
      transform.GetComponent<TextMeshPro>().enabled = false;
   }
}
```

4. KODA ZA WEBSOCKET

```
import asyncio
import websockets
from websocket import create_connection
import json
import socket
import atexit
import time

s = socket.socket()
ws = create_connection("ws://localhost:15881/v2/feedbacks")

def povezava(s):
    host = '172.20.10.9'
    port = 42424
    s.connect((host, port))
    print("Povezava vzpostavljena")
povezava(s)

async def server(websocket, path):
    while True:
        try:
            data = await websocket.recv()
            data = json.loads(data.replace("'", '"'))
            if 'Submit' in data:
                status = 'a'
                s.send(status.encode())
            else:
                off = 'b'
                s.send(off.encode())
            print(data)
        except Exception as e:
            print(e)

start_server = websockets.serve(server, "localhost", 15881)
asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
def exit_handler():
    atexit.register(s.close)
```

5. KODA ZA RASPBERRY PI

```
import socket
import RPi.GPIO as GPIO
import time
import atexit

GPIO.setmode(GPIO.BCM)
GPIO.setup(22, GPIO.OUT)

def Main():
    host = '172.20.10.9'
    port = 42424
    s = socket.socket()
    s.bind((host, port))
    s.listen(1)
    c, addr = s.accept()
    while True:
        data = c.recv(1024).decode()
        print(data)
        if 'a' in data:
            GPIO.output(22, GPIO.HIGH)
            time.sleep(0.4)
            GPIO.output(22, GPIO.LOW)
        if not data:
            break
        data = str(data).upper()
    c.close()
if__name__ == '__main':
    Main()
def exit_handler():
    atexit.register(c.close)
```

6. POVEZAVA ARDUINO IN KODA COUNTER

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using System.IO.Ports;
using System.Threading;

public class Counter : MonoBehaviour
{
   public int counter = 0;
   public string portName = "COM3";
   public CounterVzadi CounterVzadi;
   SerialPort arduino;

   void Start()
   {

      arduino = new SerialPort(portName, 9600);
      arduino.Open();
      arduino.Write("a");

   }
   public void Hit()
   {
      counter++;
      transform.GetComponent<TextMeshPro>().SetText((counter).ToString());
      CounterVzadi.UpdateVzadi(counter);
   }
   private void OnApplicationQuit()
   {
      arduino.Write("c");
      Debug.Log(counter.ToString());
   }
  }
```

7. KODA ARDUINO

```
const int Vent1= 13;
char zacni = 'c';

void setup() {
  pinMode(Vent1, OUTPUT);
  Serial.begin(9600);
  digitalWrite(Vent1, HIGH);

}


  void loop()
  {
    if (Serial.available() > 0)
    {
    zacni = Serial.read();
    while (zacni == 'a')
    {
      digitalWrite(Vent1, LOW);
      delay(19000);
      digitalWrite(Vent1, HIGH);
      delay(19000);
      zacni = Serial.read();
     }
    if (zacni == 'c')
   {
      digitalWrite(Vent1, HIGH);
   }
    }
  }
```