SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6343

Proširenje 3D interaktivne računalne igre potporom za dubinsku kameru s detekcijom pokreta

Nikolina Motočić

Zagreb, lipanj, 2019

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 13. ožujka 2019.

ZAVRŠNI ZADATAK br. 6343

Pristupnik: **Nikolina Motočić (0036493810)**
Studij: Računarstvo
Modul: Programsko inženjerstvo i informacijski sustavi

Zadatak: **Proširenje 3D interaktivne računalne igre potporom za dubinsku kameru s detekcijom pokreta**

Opis zadatka:

Vaš zadatak je proširiti raniju inačicu 3D interaktivne računalne igre "Pazi zid" razvijene primjenom programskog okvira Unity te Kinect SDK unutar preddiplomskog projekta. Proširenje treba obuhvatiti potporu za dubinsku kameru s detekcijom pokreta novije generacije Intel RealSense te unaprijeđeni sadržaj igre prema vlastitoj ideji (primjerice, vizualni elementi scene ili korisnička interakcija i zadaci). Ukratko opišite koncepciju i tehnologije korištene u razvoju postojeće aplikacije "Pazi zid" te osmislite i razradite zadano proširenje, odnosno, vlastitu ideju sadržajnih proširenja. Programski izvedite predložena proširenja te ocijenite kvalitetu nove inačice igre s motrišta iskustvene kvalitete.
Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 14. lipnja 2019.

Mentor:
Prof. dr. sc. Maja Matijašević

Djelovođa:
Doc. dr. sc. Mirjana Domazet-Lošo

Predsjednik odbora za završni rad modula:
Izv. prof. dr. sc. Ivica Botički

Sadržaj

1.Uvod

Detekcija pokreta je proces otkrivanja promjene pozicije objekta u odnosu na okolinu ili okoline u odnosu na objekt. Pokret može biti otkriven pomoću infracrvenih senzora (aktivni ili pasivni), optičkih senzora (video nadzor), zvuka, vibracija (potresi) ili magnetizma. Senzori za detekciju pokreta koriste se svakodnevno u raznim situacijama – primjeri uključuju pokretna vrata, sustave za sigurnost, rasvjetu, slavine i drugo.

U industriji video igara, koristi se posebna vrsta upravljača kako bi se pratio pokret. Takvi upravljači u sebi sadrže brzinomjer kako bi otkrili okvirnu poziciju, orijentaciju i brzinu. Popularizirala ih je tvrtka *Nintendo* 2006 godine kada je razvila svoju konzolu pod nazivom *Wii*. Ubrzo su i ostale tvrtke počele razvijati svoje upravljače i ugrađivati senzore za detekciju pokreta. Uz *Nintendo Wii* upravljače, bitno je spomenuti i Microsoft *Kinect* koji koristi infracrveni senzor u kombinaciji s računalnim vidom te *Hydru* tvrtke Razer koja koristi magnetsko polje kako bi odredila orijentaciju i poziciju upravljača.

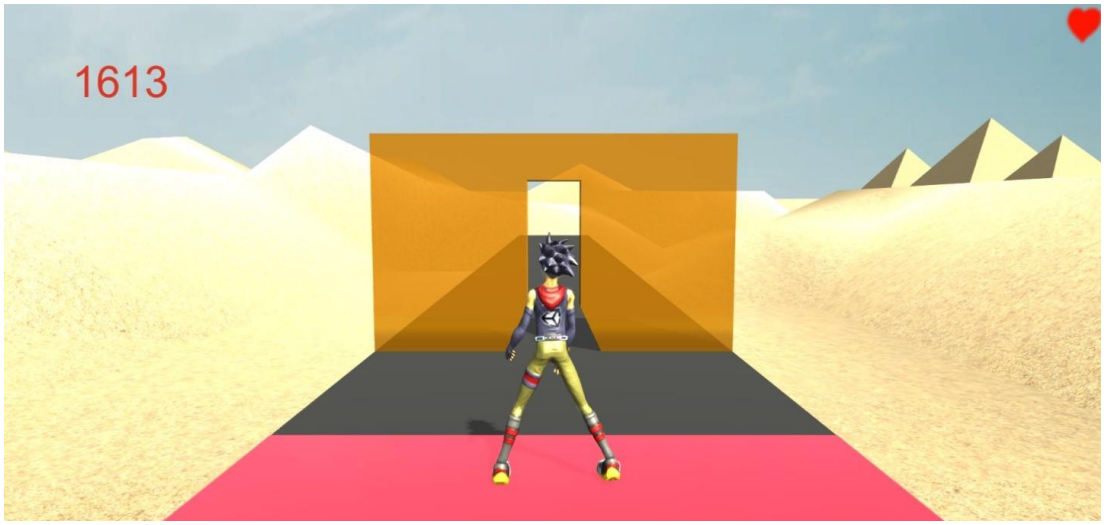
Zadatak ovog rada je proširenje ranije razvijene 3D interaktivne računalne igre „Pazi zid“, koje uključuje zamjenu upravljača Microsoft *Kinect v2* s dubinskom kamerom tvrtke Intel te razvoj programske podrške pomoću posredničke aplikacije NuitrackSDK.

Ovaj rad podijeljen je u 4 poglavlja. Nakon uvoda, u 2. poglavlju opisano je početno stanje te pregled korištene tehnologije. Kroz 3. poglavlje opisan je postupak korištenja posredničke aplikacije NuitrackSDK za praćenje pokreta korisnika. U 4. poglavlju opisane su promjene u odnosu na prethodnu verziju igre.

2. Opis početnog stanja i korištenih tehnologija

U sklopu predmeta „Projekt iz programske potpore“ u akademskoj godini 2018/2019 razvijena je interaktivna računalna igra pomoću programa Unity i Microsoftovog Kinecta v2. Kinect v2 je upravljač za video igre baziran na kameri i detekciji pokreta pomoću infracrvenog skeniranja.

Igra razvijena unutar projekta inspirirana je japanskom TV-igrom „Pazi zid“ – glavna zadaća igre je postaviti tijelo tako da se prođe kroz rupu u zidu koji se približava igraču. Ako igrač uspije proći kroz rupu u zidu, nagrađen je bodovima, a ako ne, oduzima mu se „život“. Igra završava nakon što broj „života“ padne na nulu.



Slika 1 Prikaz igre - preuzeto iz tehničke dokumenatcije [2]

Igra se sastoji od dvije glavne komponente – glavnog izbornika i same igre. Scene igre sastoje se od staze na kojoj se nalaze model-avatar i zidovi te okoline koja se može odabrati u glavnom izborniku. U glavnom izborniku odabiru se broj života, težina igre te okolina u kojoj će se igrati (moguće opcije su plaža ili pustinja). Pri završetku igre bilježi se rezultat koji se ispisuje u rang-listi igrača (engl. *leaderboard*) u dijelu glavnog izbornika.

2.1 Korištene tehnologije

Proširenje igre razvija se pomoću programskog alata Unity i posredničke aplikacije NuitrackSDK. Logika igre izvedena je pomoću skripti napisanih u programskom jeziku C#. Kao ulaz podataka o pokretima koristi se Intel RealSense D435i kamera.

Unity

Unity [3] je platforma za razvoj računalnih igara koju je razvila tvrtka Unity Techonologies i objavila u lipnju 2005. godine. Platforma se koristi za razvoj 2D i 3D igara te za simulacije za preko 25 digitalnih platformi. Unity nam omogućava unošenje vlastitih modela, zvukova, grafike, animacija, specijalnih efekata i drugih elemenata kako bi igra bila jedinstvena i po željama korisnika. Logika igre izvodi se u skriptama pisanim primarno u programskom jeziku C#, no moguće je koristiti i druge skripte programske jezike [4].

NuitrackSDK[1]

NuitrackSDK [5] je trodimenzionalni *middleware* koji je razvila tvrtka 3DiVi kao rješenje za praćenje kostura i praćenje gesti kako bi se omogućila prirodnija interakcija računala i čovjeka (eng. Natural user interface). Razvijen je za više jezika i različite platforme (Windows, Android, Linux...) . Glavna zadaća je komunikacija s trodimenzionalnim senzorima.

Intel RealSense D435i kamera

RealSense D435i kamera [6] je dubinska kamera koju je razvila tvrtka Intel kao poboljšanje na prethodni model, D435. Ona kombinira dubinsko očitavanje prethodnog modela i novo, inercijsku mjernu jedinicu (eng. *inertial mesurement unit*, *IMU*) . Dodavanjem takve jedinice omogućuje poboljšanu svijest o dubini u situacijama u kojima se kamera pomiče. U ovome projektu, koristimo modul za praćenje pokreta, samo jedan mali dio mogućnosti koju ovakve kamere pružaju.

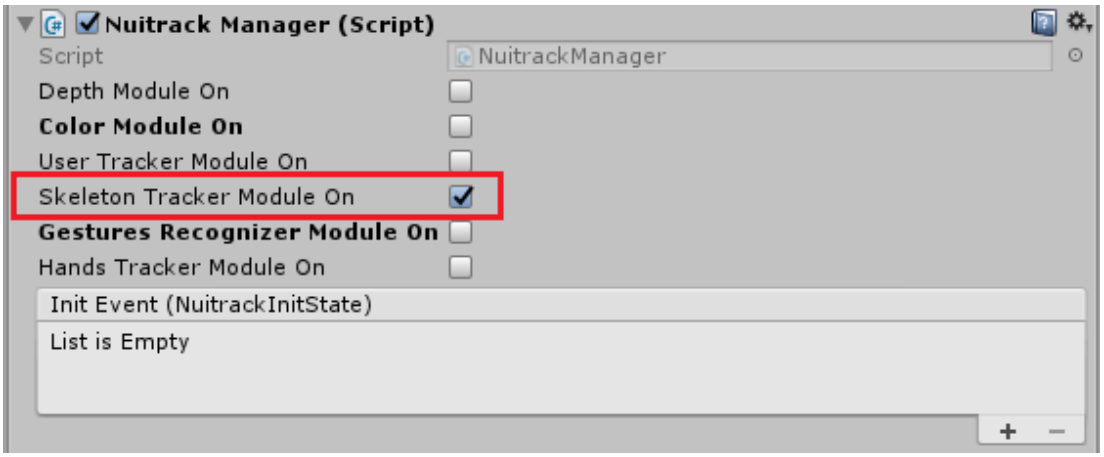
[1] Projekt i instalacija Nuitracka moraju se nalaziti na istom disku.

3. Detekcija pokreta

NuitrackSDK razvijen je kako bi se omogućila što prirodnija interakcija računala i čovjeka. Kako bi to bilo moguće, potrebna je kamera s 3D senzorom. Kamera snima prostor ispred sebe te radi detekciju korisnika. Kako bi senzor detektirao korisnika u vidnome polju, on zapravo traži zglobove korisnika. NuitrackSDK nudi praćenje 19 zglobova (Slika 1.) na cijelome tijelu te praćenje 6 gesti: zatvaranje šake, pomak šake gore, dolje, lijevo i desno i približavanje odnosno udaljavanje šake od senzora.

Slika 2 Položaj zglobova koje prati Nuitrack

Prilikom korištenja NuitrackSDKa potrebno je odabrati module koji se žele pratiti u *Nuitrack Manager* skripti(Slika 2) :



Slika 3 Modul za praćenje kostura

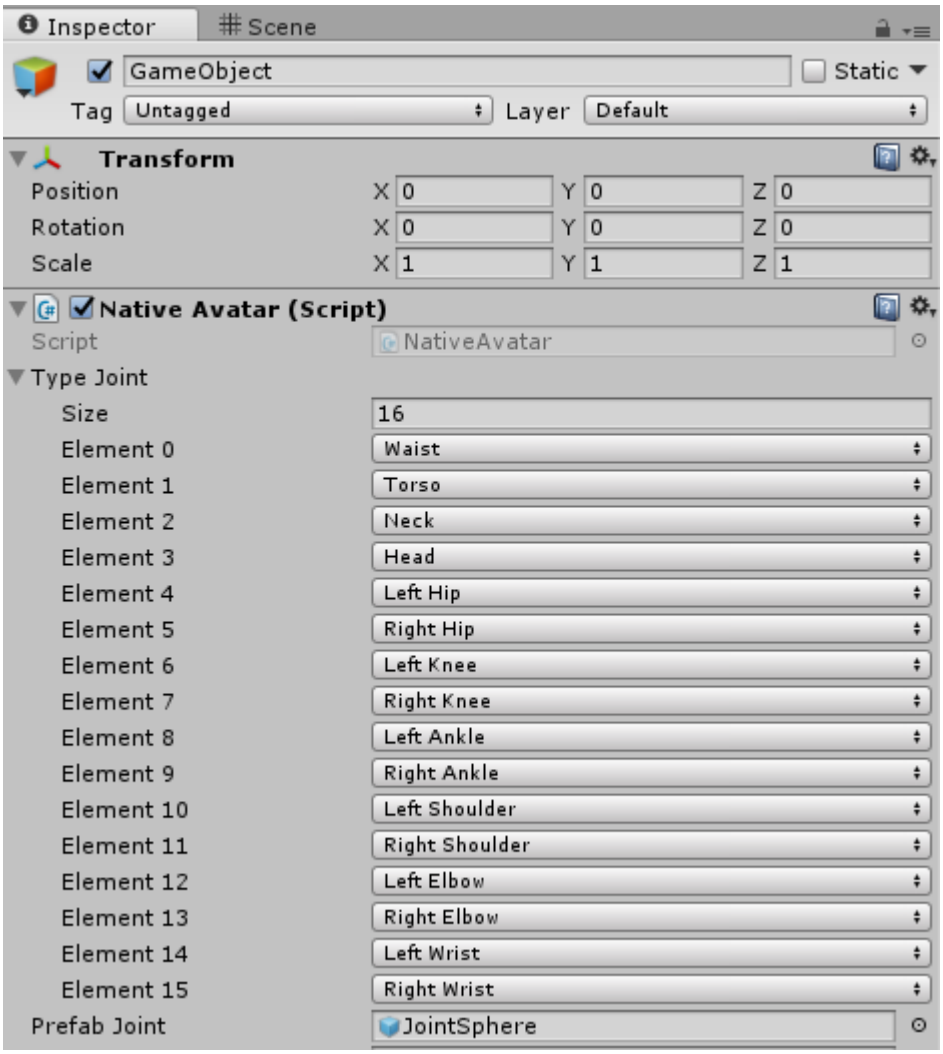
Kako bi se pratio kostur potrebno je uključiti *Skeleton Tracker Module*. Modul može pratiti 6 kostura, no zadano je da prati maksimalno 2 korisnika paralelno. Modul se sastoji od podatkovne strukture koja služi kao popis zglobova. Podaci o kosturu spremaju se u strukturi *Skeleton*. Struktura ima dvije članske varijable: *id* tipa *int* (identifikator korisnika kojeg prati) i *joints* tipa *JointType[]* gdje pamti zglobove.

Također, struktura pamti i orijentaciju zglobova koja je predstavljena matricom rotacije.

3.1 Prikaz kostura na ekranu

Kada senzor pronađe korisnika, u skripti *NativeAvatar.cs,* koja služi za prikaz avatara na ekranu, postavljamo uvjet *CurrentUserTracker.CurrentUser != 0.* Na ekranu se pojavljuje poruka o uspješnosti pronalaska korisnika.

Skripta u sebi sadrži polje *nuitrack.JointType* za pohranu informacija o zglobovima. Unutar Unityja može se specificirati zglobove koji se koriste (Slika 3)



Slika 4 Specifikacija zglobova

Kako bi se prikazao kostur na ekranu, potrebno je instancirati objekt kojim će se prikazati zglobove

```
void Start()
{
    CreatedJoint = new GameObject[typeJoint.Length];
    for (int q = 0; q < typeJoint.Length; q++)
    {
        CreatedJoint[q] = Instantiate(PrefabJoint);
        CreatedJoint[q].transform.SetParent(transform);
    }
    message = "Skeleton created";
}
```

Isječak 1 Instanciranje objekta za prikaz zblgoova

Izvođenjem ovog koda dolazi do prikaza točaka na ekranu, no sve točke će biti na istom mjestu – u središtu kostura. Kako bi se prikazali zglobovi onako kako ih senzor očitava, potrebno je postaviti poziciju objekta na poziciju zgloba, te ju smanjiti množenjem s konstantom 0,001[2] kako bi se metri pretvorili u milimetre.

```
Vector3 newPosition = 0.001f * joint.ToVector3()
```

Isječak 2 Promjena pozicije objekta koji prikazuje zglob

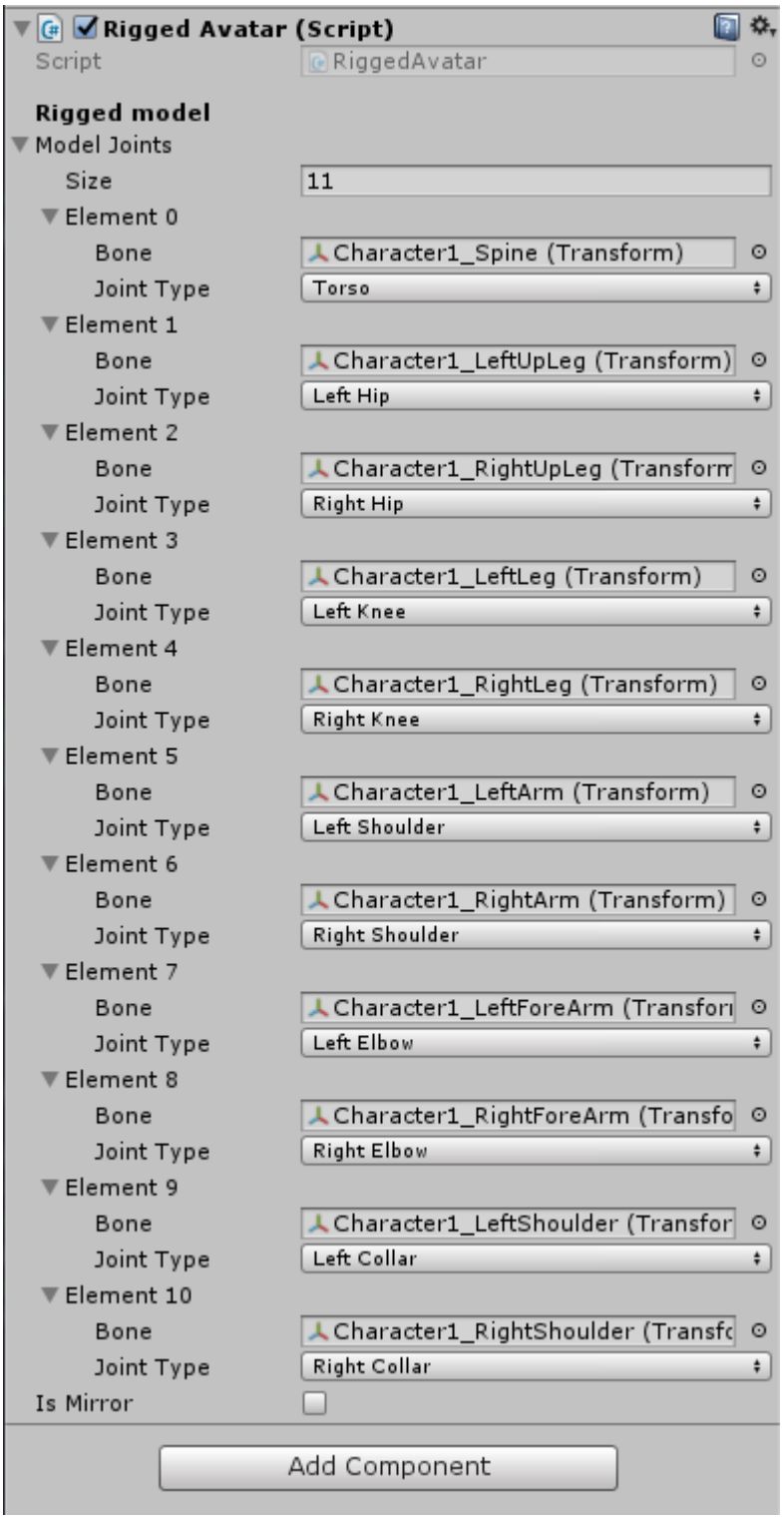
3.2 Mapiranje na 3D model

Praćenje kostura, tj detekciju pokreta može se prikazati i na drugi način – pomoću čovjekolikog 3D modela. Prilikom izrade čovjekolikog 3D modela potrebno je izraditi (pojednostavljeni) „kostur“ koji se sastoji od segmenata međusobno povezanih zglobovima. Takav kostur može se povezati sa senzorom, odnosno pomoću NuitrackSDK izvesti kontroliranje 3D modela pokretom. Potrebno je odabrati jednostavan 3D model jer neke kompleksnije modele će biti vrlo teško mapirati i prilagoditi. Model se može mapirati (povezati detektirane zglobove i zglobove kostura 3D modela) na dva načina: indirektno i direktno.

3.2.1. Indirektno mapiranje

Kod indirektnog mapiranja, pozicija modela računa se prema poziciji kralježnice modela. U skripti koja je zadužena za povezivanje modela i kostura definirano je polje zglobova *modelJoints*. U *Inspector* prozoru u Unityju zadaje se točan tip zgloba zglobovima modela koji je naveden u popisu strukture *Skeleton*. (Slika 4)

[2] U Unityju, prikaz jedne jedinice odgovara u stvarnosti jednom metru stoga moramo pretvoriti udaljenost objekata iz metra u milimetre

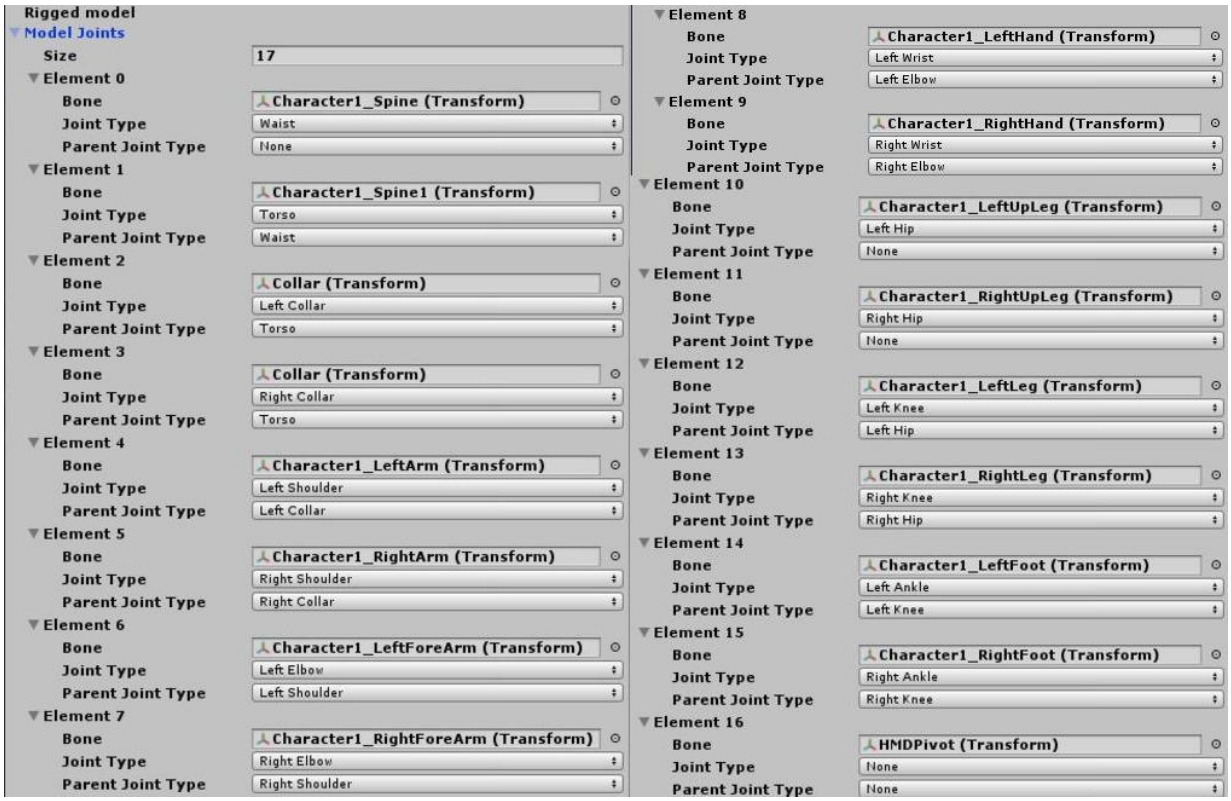


Slika 5 Mapiranje zglobova

Pri pokretanju skripte, povezivanje dobivenog zgloba (*modelJoint)* s tipom zgloba (*nuitrack.JointType)* sprema se u riječnik *jointsRigged.* Zbog načina interpretacije podataka sa senzora, potrebno je provjeriti rotaciju zglobova i po potrebi zrcaliti tako da model izvodi isti pokret kao i korisnik.

3.2.2. Direktno mapiranje

Kod direktnog mapiranja bitna je udaljenost između zglobova, odnosno između zgloba-roditelj i zgloba-dijete (na primjer lakat – zglob šake). Za razliku od indirektnog mapiranja, za direktno mapiranje potreban je veći broj zglobova. Kod ovakvog načina mapiranja, svaki zglob je neovisan o drugima te se moraju definirati ovisnost *roditelj-dijete* za svaki čvor. (Slika 5)



Slika 6 Direktno mapiranje zglobova 3D modela

Zbog te neovisnosti, uz poziciju torza (kralježnice) mora se znati i pozicija ostalih zglobova. Potrebno je povezati zglobove „linijama“ koje zapravo predstavljaju kosti. Također, potrebno je definirati i minimalnu udaljenost između zglobova, inače dolazi do deformacije modela u sceni.

```
Vector3 newPos = Quaternion.Euler(0f, 180f, 0f) * (0.001f * joint.ToVector3());
modelJoint.bone.position = newPos;
```

Isječak 3 Povezivanje zglobova

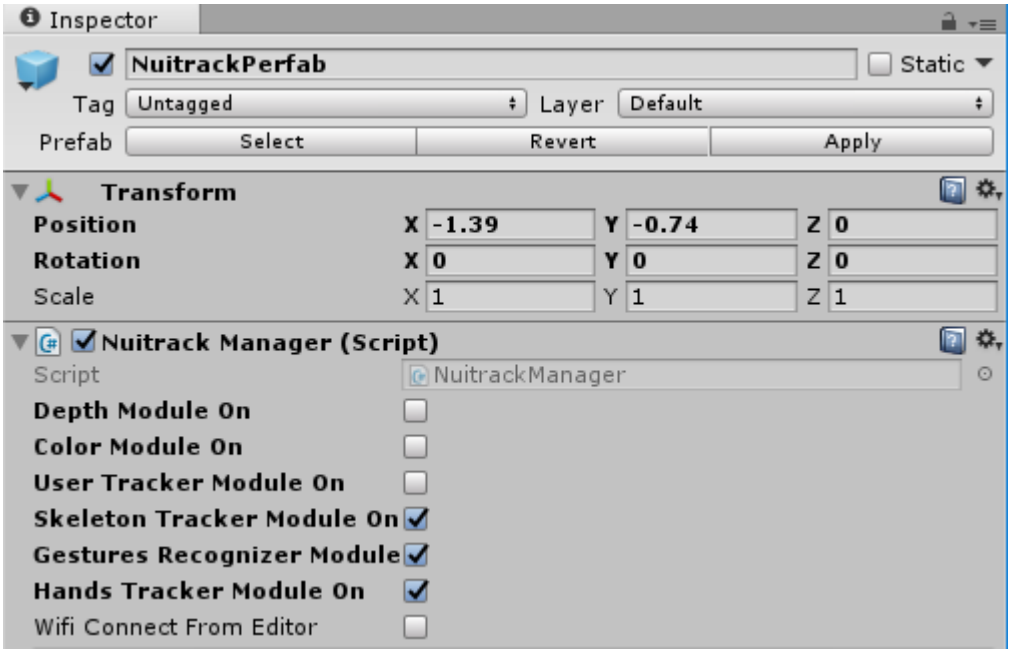
Na kraju, potrebno je skalirati „kosti“, odnosno udaljenosti između zglobova. Skaliranje će se odvijati dinamički te će se prilagođavati korisniku. Za najbolji

rezultat mapiranja, preporučuje se korištenje modela tjelesnih proporcija sličnih korisniku kako ne bi dolazilo do deformacije modela.

3.3. Geste

NuitrackSDK nudi mogućnost praćenja gesti. Konkretno, može detektirati 6 gesti: zatvaranje šake, pomak šake gore, dolje, lijevo i desno, te približavanje i udaljavanje šake (eng. *push*). Geste se mogu odvijati jednom ili oba dlana istovremeno (eng. *multi-touch)*.

Ako želimo koristiti geste, uključuju se moduli za praćenje dlanova *Nuitrack Hand Tracker* i *Nuitrack Gesture Recognizer*, te modul *Nuitrack Skeleton Tracker* (Slika 6)



Slika 7 Modul za korištenje gesti

Modul *Nuitrack Hand Tracker* zadužen je za praćenje dlanova. Sadrži dvije strukture koje pamte podatke o poziciji dlanova, te radi li se o lijevom ili desnom dlanu. Struktura *Hand* također pamti koliko je dlan zatvoren što je korisno prilikom izrade rukom upravljanih izbornika kako bi se izbjegli problem „fantomskih klikova“ (klikovi koji su registrirani kao takvi, iako to nije bila korisnikova namjera). Modul provjerava o kojem se dlanu radi. Ako

senzor ne vidi ruku, članska varijabla *active* postavlja se na *false* te je taj dlan „skriven“.

```
private void NuitrackManager_onHandsTrackerUpdate(nuitrack.HandTrackerDatahandTrackerData)
{
    bool active = false;
    bool press = false;
    foreach (nuitrack.UserHands userHands in handTrackerData.UsersHands)
    {
        if (currentHand == Hands.right && userHands.RightHand != null)
        {
            baseRect.anchoredPosition = new Vector2(userHands.RightHand.Value.X * Screen.width,
                                                    -userHands.RightHand.Value.Y * Screen.height);
            active = true;
            press = userHands.RightHand.Value.Click;
        }
        else if (currentHand == Hands.left && userHands.LeftHand != null)
        {
            baseRect.anchoredPosition = new Vector2(userHands.LeftHand.Value.X * Screen.width,
                                                    -userHands.LeftHand.Value.Y * Screen.height);
            active = true;
            press = userHands.LeftHand.Value.Click;
        }
    }
}
```

Isječak 4 Funkcija za praćenje dlana

Kako bi se lakše promatralo što se događa, dlanove prikazujemo pomoću ikona. Izgled ikone ovisit će o tome je li dlan otvoren ili zatvoren. Modul sam po sebi može pratiti gestu zatvaranja dlana i registrirati ju kao funkciju *click*. Želi li se koristiti neku od preostalih gesti, potrebao je modul *Nuitrack Gesture Gecognizer*.

Modul *Nuitrack Gesture Recognizer* zadužen je za prepoznavanje gesti. Stanje geste izraženo je numeričkom vrijednošću koja odgovara postotku izvršenosti geste. Prilikom korištenja gesti, potrebno je točno definirati koja se gesta koristi.

```
private void NuitrackManager_onNewGesture(nuitrack.Gesture gesture)
{
    switch (currentViewMode)
    {
        case ViewMode.Preview:

            if (gesture.Type == nuitrack.GestureType.GestureSwipeLeft)
            currentPage = Mathf.Clamp(++currentPage, 0, numberOfPages);
            if (gesture.Type == nuitrack.GestureType.GestureSwipeRight)
            currentPage = Mathf.Clamp(--currentPage, 0, numberOfPages);
        break;
    }
}
```

Isječak 5 Definiranje gesti

Kod korištenja svih gesti može doći do pojave „fantomskih klikova“. To se događa prilikom korištenja više gesti istovremeno. Na takve klikove utječe brzina radnje. Na primjer, ako koristimo kombinaciju gesti zatvaranja šake i pomicanja šake desno, zatvaranje šake mora biti sporije od njenog pomicanja desno kako bi se gesta registrirala kao obavljena. [3]

```
else if (selectedButton != null)
{
    if (press)
    {
        if (eventData.delta.sqrMagnitude < dragSensitivity)
        {
            eventData.dragging = true;
            selectedButton.OnPointerDown(eventData);
        }
    }
    else if (eventData.dragging)
    {
        eventData.dragging = false;
        selectedButton.OnPointerUp(eventData);
    }
}
```

Isječak 6 Primjer rješavanja problema fantomskih klikova

[3] Isječak preuzet s (5)

4. Oblikovanje proširenja postojećeg rješenja

Kao posljedica promjene upravljača, nužno je provesti dvije promjene – izabrati novi model koji će predstavljati igrača te prilagodba glavnog izbornika na nove geste.

4.1 Novi model

Stari model korišten u prethodnoj verziji igre pretjerano je složen za način na koji NuitrackSDK mapira i sprema podatke o kosturu. Model ima više kostiju te su neke kosti drugačije smještene u odnosu na očitavanje posredničke aplikacije. Zbog toga prilikom mapiranja dolazi do deformacije avatara.

Odabran je jednostavniji model s manjim brojem kostiju. Kosti novog modela se u potpunosti poklapaju s kostima koje očitava NuitrackSDK.

Slika 8 Novi model

Model je mapiran indirektno prema ranije opisanom postupku. Korišteni model besplatan je i preuzet s *Asset Storea* programa Unity. [8]

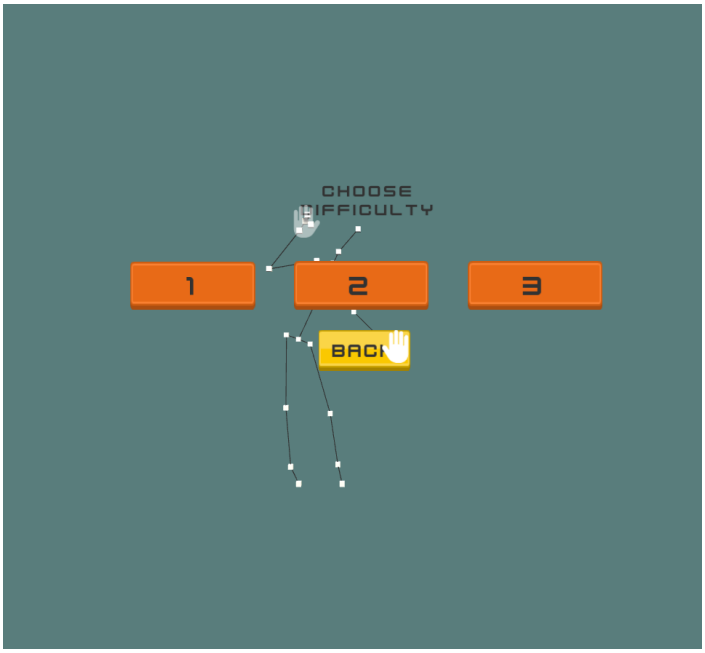
Cilj igre je proći kroz rupu u zidu. Ako igrač ne prođe kroz rupu, gubi "život". Kako bi znali je li igrač prošao kroz rupu ili ne, potrebno je detektirati sudar igrača i zida. Detekciju sudara provjerava skripta *HitSpheres.cs*. Prilikom pokretanja igre, u metodi *CreateHitSpheres()* skripte *PlayerStats.cs*, na modelu se stvaraju transparentni cilindri koji svojom veličinom i orijentacijom odgovaraju kostima modela. Cilindri se pozicioniraju prema kostima modela, te se u metodi *positionSpheres()* repozicioniraju prema poziciji avatara. Ako dođe do sudara cilindra i zida, članska varijabla *hasCollided* postaje true te se pokreće metoda za trenutno uništavanje zida i brisanje života.

```
void positionSpheres()
    {
        int i = 0;
        foreach (HumanBodyBones bone in System.Enum.GetValues(typeof(HumanBodyBones)))
        {
            if (bone == HumanBodyBones.Jaw) continue;
            if (bone == HumanBodyBones.UpperChest) return;
            if (GetComponent<Animator>().GetBoneTransform(bone) != null)
                        { hitspheres[i].transform.position = GetComponent<Animator>().GetBoneTransform(bone).position; }
            i++;
        }
    }
```

Isječak 7 Pozicioniranje cilindra

4.2 Prilagodba glavnog izbornika

U glavnom izborniku prethodne verzije koristila se gesta *push,* te se na ekranu prikazivao kostur.



Slika 9 Glavni izbornik prethodne verzije - preuzeto iz [5]

U novoj verziji, desnim dlanom se upravlja pokazivačem miša. Gestom zatvaranja šake generira se klik miša.

```
if (pressed != press)
{
    pressed = press;

    if (pressed)
    {
        MouseOperations.MouseEvent(MouseOperations.MouseEventFlags.LeftUp | MouseOperations.MouseEventFlags.LeftDown);
    }
}
```

Isječak 8 Postavljanje zastavice za pritisak lijeve tipke

Na pozadinu izbornika dodana je skripta *HandCursor.cs* u kojoj se nalazi metoda za obradu događaja *NuitrackManager_onHandsTrackerUpdate* i klasa *Mouse Operation* koja definira operacija miša. Miš ima 8 operacija – pritisak ili otpuštanje lijeve, desne ili srednje tipke, pomicanje miša i mirovanje. Ako želimo generirati klik, potrebno je pritisnuti lijevu tipku miša. Kako bi povezali gestu zatvaranja šake s generiranjem klika potrebno je postaviti zastavicu koja pamti pritisak lijeve tipke miša.

Ovime su zadana proširenja uspješno izvedena.

5. Zaključak

Cilj rada bio je proširiti postojeću igru „Pazi zid“ promjenom upravljača, što je uspješno ostvareno. Novi upravljač je Intel RealSense kamera. Kroz rad proučena je detekcija pokreta i način na koji nova kamera i posrednička aplikacija NuitrackSDK koriste dobivene podatke o modelu kostura korisnika. Proučeno je i upravljanje glavnog izbornika gestama. Programski je izvedeno mapiranje novog modela-avatara i proširenje glavnog izbornika. U glavnom izborniku koriste se nove geste zatvaranja šake dok je prethodna verzija koristila gestu *push*. Novi model kostura jednostavniji je od prethodnog po broju i razmještaju kostiju.

Popis literature

1. Dent, S. „Wave goodbye to Microsoft's original Kinect for Windows“, https://www.engadget.com/2014/12/31/oroginal-kinect-discontinued/, prosinac 2014 g.

2. Glavaš D., Kudoić N., Motočić N., Jelović M., Ramljak A., Škrabo I., Kroflin D., Poglar M. „Razvoj interaktivne računalne igre pomoću Kinecta“, tehnička dokumentacija projektnog zadatka izrađenog u okviru predmeta Projekt iz programske potpore, Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, ak. god. 2018./2019.

3. Unity: https://unity3d.com/unity

4. Programski jezici u Unityju : https://en.wikipedia.org/wiki/Unity_(game_engine)#Overview

5. Nuitrack SDK, službena dokumentacija https://download.3divi.com/Nuitrack/doc/pages.html

6. „Izrada galerije pomoću gesti“ , NuitrackSDK, službena dokumentacija, https://download.3divi.com/Nuitrack/doc/UnityGallery_page.html

7. Intel RealSense dubinska kamera: https://www.intelrealsense.com/depth-camera-d435i/

8. FlightUnit, ntny, “Unity-chan!”, 3D model: https://assetstore.unity.com/packages/3d/characters/unity-chan-model-18705

Sažetak

Proširenje 3D interaktivne računalne igre potporom za dubinsku kameru s detekcijom pokreta

U radu je proširena ranija inačica 3D interaktivne računalne igre „Pazi zid“ razvijene u okviru preddiplomskog projekta. Proširenje se temelji na potpori za Intelovu dubinsku kameru s detekcijom pokreta novije generacije pod nazivom RealSense. U radu je opisan postupak kojime je ostvareno zadano proširenje osmišljeno i programski izvedeno korištenjem programskog okvira Unity te posredničke aplikacije NuitrackSDK.

Ključne riječi: Detekcija pokreta; dubinska kamera; računalna igra; Unity; NuitrackSDK

Extension of a 3D interactive computer game using a depth camera with movement detection

This thesis presents an extension of an earlier version of a 3D interactive computer game „Hole in the Wall“ that was developed as an undergraduate project. The extenstion includes the support for a newer generation Inter RealSense depth camera with movement detection. The thesis describes are the necessary steps to design and implement the assigned extension by using the Unity and NuitrackSDK middleware.

Keywords: Motion detection; depth camera; computer game; Unity; NuitrackSDK