SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Aplikacija za praćenje staze sječiva kod presijecanja 3D objekata u virtualnoj stvarnosti

Dominik Juršić

Zagreb, svibanj 2024.

Sadržaj

Uvod

Cilj rada je izgraditi aplikaciju koja omogućuje praćenje i izvoz podataka o putanji alata za sječenje objekata. Aplikacija omogućuje rezanje objekata u virtualnoj stvarnosti. Tijekom rezanja objekata spremaju se podatci o relativnoj poziciji i rotaciji alata s obzirom na mrežu trokuta koju želimo presjeći ili početnu poziciju sječiva. Aplikacija omogućuje izvoz tih podataka. Rezanje objekata obavlja se transformiranjem objekta u 3D polje podataka (voxele) i mijenjanjem njihovih vrijednosti. Objekti koji se mogu transformirati reprezentirani su zatvorenom mrežom trokuta. Iz te mreže trokuta izgradi se 3D polje koje reprezentira kvadar dimenzije veličine objekta po osima. To polje se ispuni točkama (voxeli) za koje se odredi jesu li unutar ili izvan objekta (0 – unutar, 1 – izvan). Takvo polje se koristi u Marching Cubes algoritmu da bi iscrtalo poligone rezanog objekta. Alat koji koristi za presijecanje objekata mijenja to polje tako da točke kojima prolazi postavlja na 0. Nakon presijecanja objekta Marching Cubes algoritam će iscrtati nove poligone promijenjenog objekta.

1. Pregled područja

Najzahtjevniji dio projekta je transformiranje i rezanje objekta. Postoje gotove implementacije za razdvajanje objekta na dva dijela s obzirom na proizvoljnu ravninu, ali one ne odgovaraju potrebi da se može zarezati parcijalno u objekt. Zato se u projektu koristi pristup gdje se iz mreže poligona objekta konstruira 3d polje gdje vrijednosti reprezentiraju je li točka unutar ili izvan objekta. Iz tog polja se onda može rekonstruirati objekt algoritmom Marching Cubes koji iterira po ćelijama 3D polja i na rubovima objekta dodaje poligone. Takav zapis i crtanje objekta nam omogućuje da rezanje objekata tako da mijenjamo vrijednosti u polju i ponovo provedemo Marching cubes kako bi nacrtali izmijenjeni objekt. Problem takvog pristupa umjesto razdvajanjem poligona s obzirom na proizvoljnu plohu u prostoru je potreba velikog broja točaka koje reprezentiraju objekt u polju.

2. Specifikacija programske potpore

2.1. Funkcionalni zahtjevi

U aplikaciji postoji samo jedan tip korisnika. Korisnik može koristiti aplikaciju kako bi u virtualnom okruženju mogao prerezati odabrane objekte. Korisnik nakon toga može izvesti podatke praćenja igrače palice u kojoj se nalazi alat za rezanje.

2.1.1. Obrasci uporabe

Opis obrazaca uporabe

UC1 – Rezanje objekata

- **Glavni sudionik:** Korisnik
- **Cilj:** Potpuno ili djelomično rezanje objekta
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
    1. Korisnik prilazi objektu
    2. Korisnik desnom palicom prolazi kroz objekt

UC2 – Pokretanje praćenja sječiva

- **Glavni sudionik:** Korisnik
- **Cilj:** Pokrenuti praćenje podataka o poziciji i rotaciji desne igraće palice
- **Preduvjet:** -
- **Opis osnovnog tijeka:**
    1. Korisnik se odlučuje za pokretanje snimanja podataka
    2. Korisnik pritišće „primary button“ na desnoj igraćoj palici

UC3 – Zaustavljanje praćenja sječiva

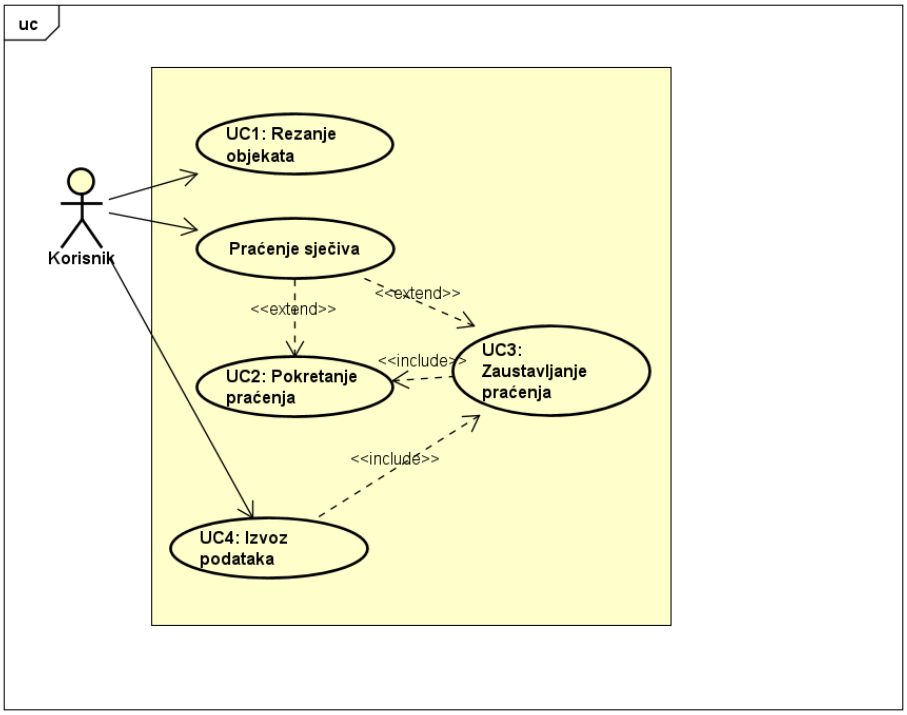
- **Glavni sudionik:** Korisnik

- **Cilj:** Zaustaviti praćenje podataka o poziciji i rotaciji desne igraće palice
- **Preduvjet:** Započeto praćenje podataka
- **Opis osnovnog tijeka:**
    1. Korisnik se odlučuje za zaustavljanje snimanja podataka
    2. Korisnik pritišće „primary button“ na desnoj igraćoj palici

UC4 – Izvoz podataka

- **Glavni sudionik:** Korisnik
- **Cilj:** Izvesti podatke o praćenju sječiva u dokument
- **Preduvjet:** Pokrenuto praćenje barem jednom
- **Opis osnovnog tijeka:**
    1. Korisnik pokreće snimanje
    2. Korisnik zaustavlja snimanje
    3. Korisnik izvozi snimljene podatke

Dijagram obrasca uporabe



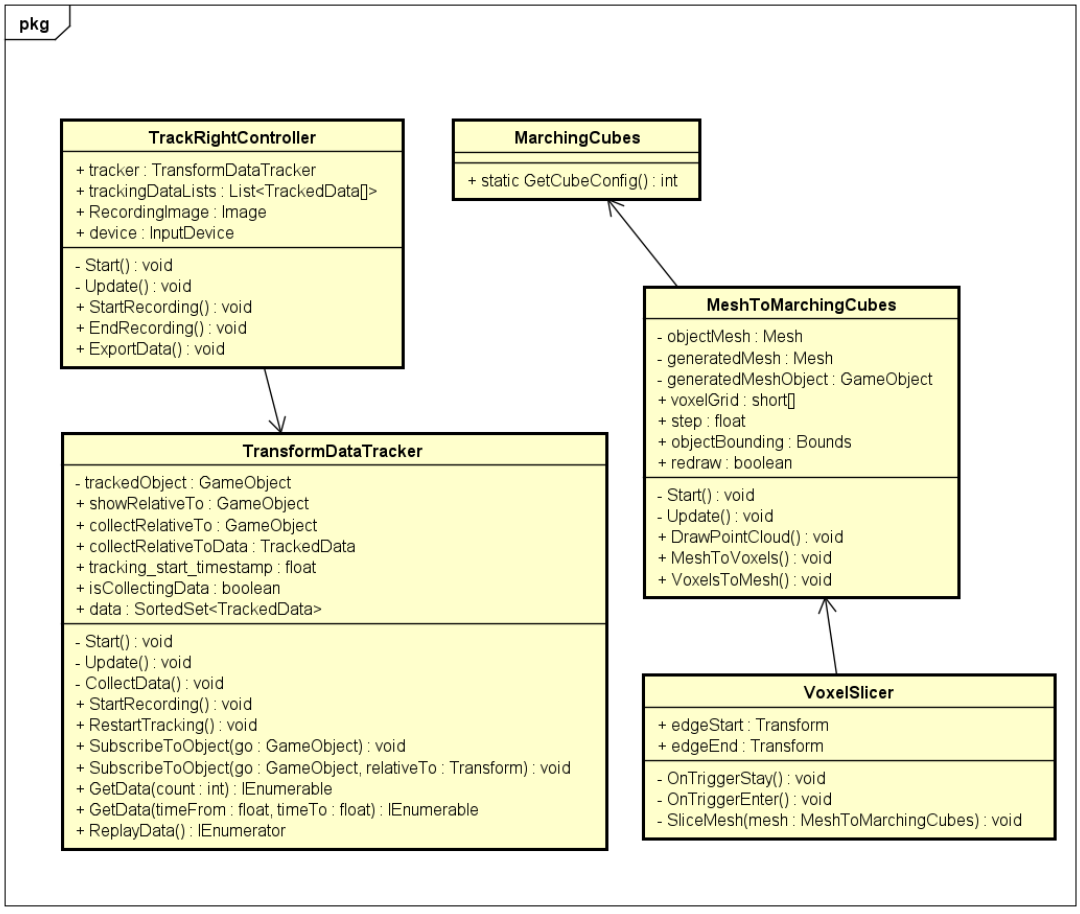
Slika 2.1 Dijagram obrasca uporabe

2.1.2. Dijagram klasa

U aplikaciji se osim Unity gotovih skripti koriste 5 vlastitih skripti. „MarchingCubes.cs“ služi za određivanje konfiguracije jedinične kocke u algoritmu. „MeshToMarchingCubes.cs“ sadrži funkcije za pretvaranje objekta u voxele i implementaciju MC algoritma za iscrtavanje objekta definiranog voxelima. „VoxelSlicer.cs“ skripta se nalazi na objektu koji se koristi za rezanje objekata i sluša za sudare između vlastitog Collidera i „MeshCollidera“ objekta koji sadrži na sebi skriptu „MarchingCubes.cs“. „TransformDataTracker.cs“ služi za spremanje podataka o praćenju objekta na kojem se nalazi u odnosu na proizvoljni objekt (odnosno Transform komponentu tog objekta). „TrackRightController.cs“ pokreće i zaustavlja snimanje „TransformDataTracker.cs“ koji se nalazi na desnoj igraćoj palici i sprema podatke svakog pokrenutog snimanja. „TrackRightController.cs“ sadrži i funkciju za izvoz podataka.

Dijagram

Sve klase nasljeđuju klasu MonoBehaviour iz Unity paketa.



Slika 2.2 - Dijagram klasa

2.2. Nefunkcionalni zahtjevi

Generiranje nove mreže poligona mora se izvesti prihvatljivom brzinom i ne smije blokirati korisnika od korištenja aplikacije.

Rezolucija zapisa objekta 3D mora biti dovoljno velika da se ne gubi oblik originalnog objekta.

3. Implementacija i korisničko sučelje

3.1. Korištene tehnologije i alati

Aplikacija je razvijena u okruženju Unity (verzija 2022.3.4f1). Za implementaciju virtualne realnosti korišten je Unity XR paket. Aplikacija je razvijana testiranjem na Meta Quest 2. Kod je pisan u jeziku C# u okruženju Visual Studio 2022.

3.2. Opis implementiranog rješenja

Ideja aplikacije je omogućiti korisniku dinamičko rezanje objekata u VR-u. Aplikacija bilježi kretanje alata ili ruke s kojom se reže objekt i omogućuje njihov izvoz kako bi se na tim podatcima mogao trenirati model strojnog učenja. Rezanje objekata ostvareno je spremanjem podataka u 3D polje (voxeli). Ti podatci su zapisano kao 0 (točka nije u objektu) ili 1 (točka je u objektu). Rezanje se ostvaruje tako da se prati oštrica i za točke objekta koje se preklapaju s oštricom se postave na 0 (izbace se iz objekta). Iz voxela se konstruira nova mreža trokuta korištenjem algoritma „Marching Cubes“. Rezultat rezanja objekta može se vidjeti na slici (Slika 3.4).

3.2.1. Računanje voxela iz mreže poligona objekta

Kako bi reprezentirali objekt s voxelima treba nam način da izračunamo koje su točke u prostoru unutar njegove mreže objekata. Za konveksne objekte to se može jednostavno odraditi provjerom orijentacije točke naprema svakom poligonu. Ako su sve točke „iznad“ ili „ispod“ ravnine u kojima se nalaze poligoni onda se točka nalazi unutar objekta („iznad“ u ovom kontekstu se definira tako da je skalarni umnožak točke u homogenim koordinatama i ravnine veći od nule, a „ispod“ manji od nule). Hoćemo li provjeravati „ispod“ ili „iznad“ ovisi o orijentaciji koordinatnog sustava objekta. Ovaj pristup za konkavne objekte neće funkcionirati jer postoje točke unutar objekta koje imaju različite orijentacije za ravnine definirane poligonima objekta.

S obzirom na to da želimo u aplikaciji koristiti i konkavne objekte trebat će nam drugi pristup. U aplikaciji je implementiran pristup praćenja zrake. Odredimo omeđujući kvadar (bounding box – dalje u tekstu BB) paralelan s osima koordinatnog sustava i odaberemo smjer zrake (jedna od osi sustava) i onda za sve točke ravnine BB-a kojima je smjer zrake normala ispucamo zraku i tražimo presjeke. Nakon prvog presjeka sve točke su unutar objekta do drugog presjeka. Zatim su sve točke do trećeg presjeka izvan, pa do četvrtog unutar i tako dalje. U implementaciji je korišten malo drugačiji pristup, ali je slična ideja. Korištenjem Unity Ray objekta i Physics.Raycast() metode filtrirano samo na objekt na kojem se nalazi skripta možemo dobiti prvu točku presjeka zrake s „MeshColliderom“ objekta (Kod 3.1). Ako je skalarni umnožak normale i smjera zrake nenegativan onda su sve točke od izvora zrake do točke sudara unutar objekta pa postavljamo vrijednost voxela na 1. Inače ako je skalarni umnožak negativan onda je poligon orijentiran prema nama pa znamo da su točke izvan objekta.

```
void MeshToVoxels()
    {
        objectBounding = objectMesh.bounds;
        Vector3Int gridSize = new Vector3Int(
            Mathf.CeilToInt(2 + objectBounding.size.x /
step),
            Mathf.CeilToInt(2 + objectBounding.size.y /
step),
            Mathf.CeilToInt(2 + objectBounding.size.z / step)
        );

        voxelGrid = new short[gridSize.x, gridSize.y,
gridSize.z];

        Vector3 localPoint = Vector3.zero;
        int pointsInMesh = 0;
        for (float x = objectBounding.min.x; x <=
objectBounding.max.x; x += step)
            for (float y = objectBounding.min.y; y <=
objectBounding.max.y; y += step)
                for (float z = objectBounding.min.z; z <=
objectBounding.max.z ; z += step)
                {
                    localPoint.Set(x, y, z);
```

```
                    Vector3 point =
transform.TransformPoint(localPoint);

                    RaycastHit[] hits =
Physics.RaycastAll(point, Vector3.up,
float.PositiveInfinity);

                    List<RaycastHit> filteredHits = new
List<RaycastHit>();
                    foreach (RaycastHit hit in hits)
                    {
                        if (hit.collider.gameObject ==
gameObject)
                        {
                            filteredHits.Add(hit);
                        }
                    }

                    int indexX = 1 + Mathf.FloorToInt((x -
objectBounding.min.x) / step);
                    int indexY = 1 + Mathf.FloorToInt((y -
objectBounding.min.y) / step);
                    int indexZ = 1 + Mathf.FloorToInt((z -
objectBounding.min.z) / step);

                    int nHits = filteredHits.Count;

                    if(nHits >= 1 &&
Vector3.Dot(filteredHits[0].normal, Vector3.up) >= 0)
                    {
                        voxelGrid[indexX, indexY, indexZ] =
1;
                        pointsInMesh++;

                    }
                }
            }
```

Kod 3.1 – Funkcija za računanje voxela iz mreže poligona objekta

Nedostatak ovakve implementacije je potreba da objekt ima „MeshCollider“. Moguće je implementirati vlastito praćenje zrake koje bi tražilo sudare s

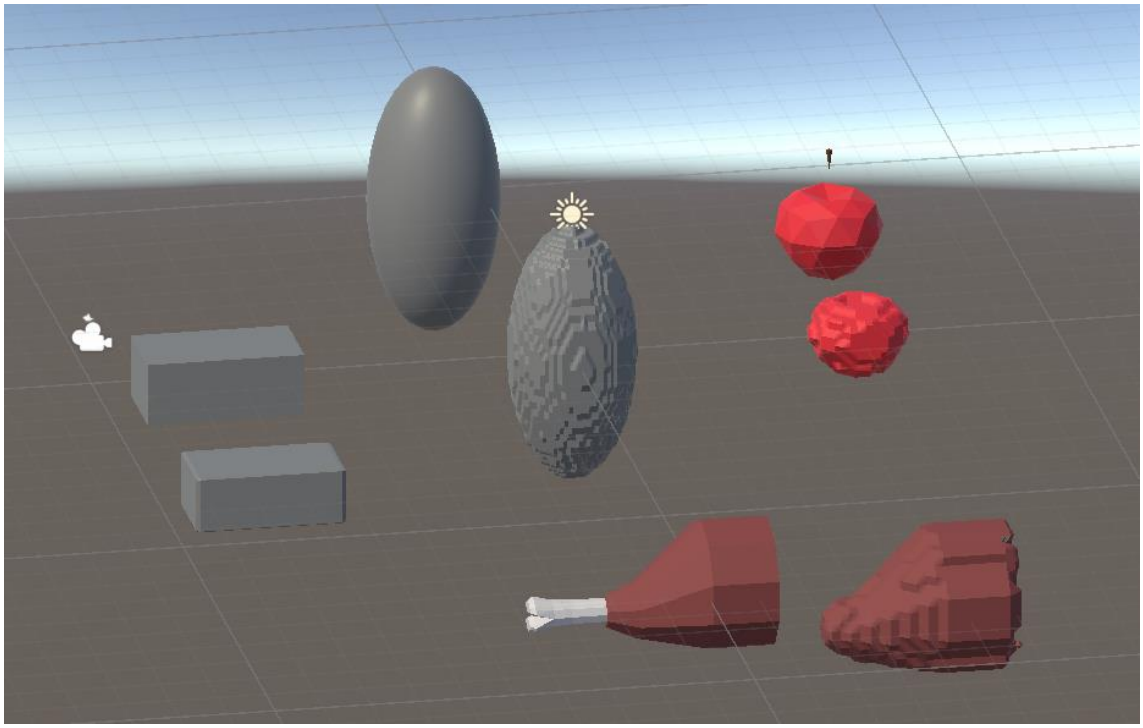
poligonima iz podataka o točkama i konstrukciji poligona u objektu. Brzina ovog algoritma nije bitna jer možemo izračunati voxele izvan aplikacije i spremiti ih i direktno koristiti u aplikaciji. Nije potrebno unutar aplikacije raditi pretvorbu.

3.2.2. Marching Cubes algoritam

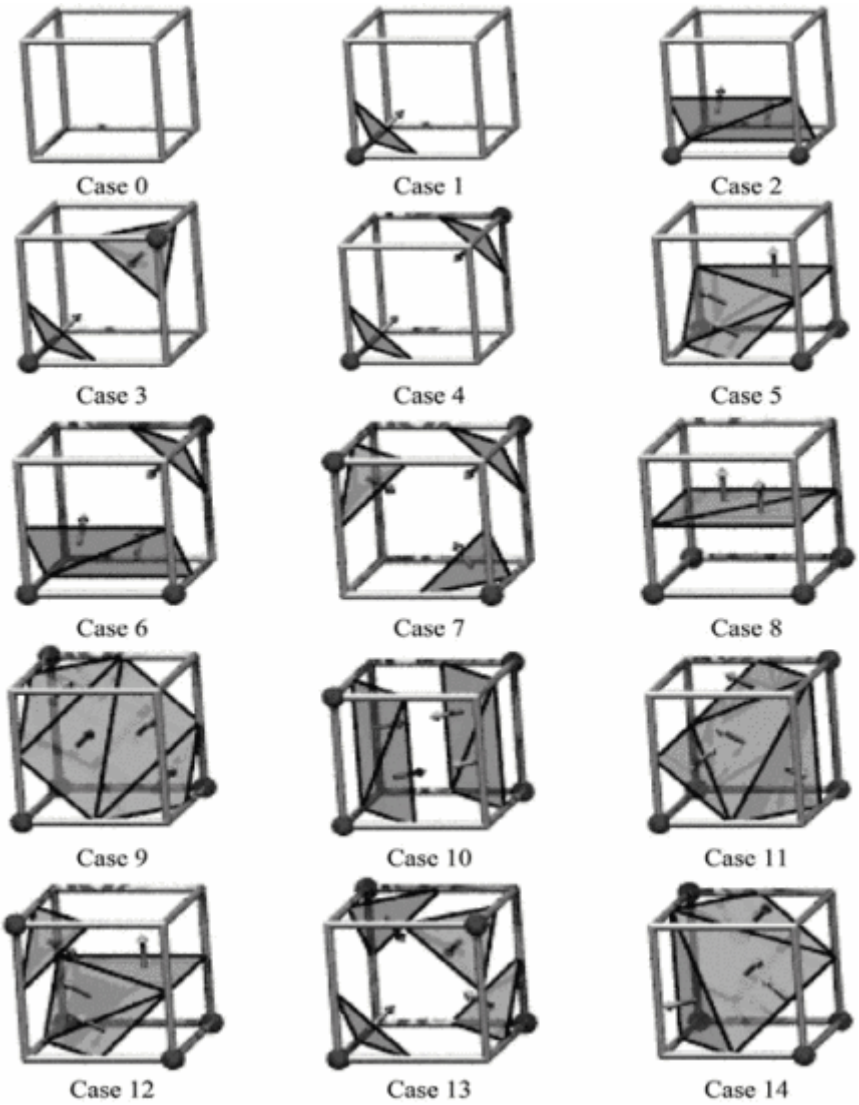
MC algoritam je metoda za konstruiranje i prikaz izopovršine [1]. Jedinična kocka prolazi po 3D polju podataka i čita vrijednosti svojih 8 vrhova. S obzirom na odabrani prag ako su vrijednosti ispod praga točka se smatra da je izvan objekta, a inače unutar. S obzirom na konfiguraciju kocke dodaju se trokuti u izopovršinu objekta. Kocka ima 256 različitih konfiguracija od kojih je 15 jedinstveno i ostale se mogu dobiti transformacijama tih 15 (Slika 3.2). Kada algoritam prošeta po cijelom prostoru izgradit će se cijela površina objekta.

U aplikaciji se nalazi skripta „MarchingCubes.cs“ koja za dane vrijednosti vrhova jedinične kocke vraća konfiguraciju (indeks konfiguracije). Taj indeks onda koristi skripta „MeshToMarchingCubes.cs“ za dodavanje potrebnih poligona u mrežu.

Rezultat transformacije nekoliko objekata u voxele pa nazad u mrežu poligona korištenjem MC se može vidjeti na slici (Slika 3.1).



Slika 3.1 - Transformacija objekta u voxele i iscrtavanje mreže MC algoritmom



Slika 3.2 Konfiguracije u MC algoritmu [1]

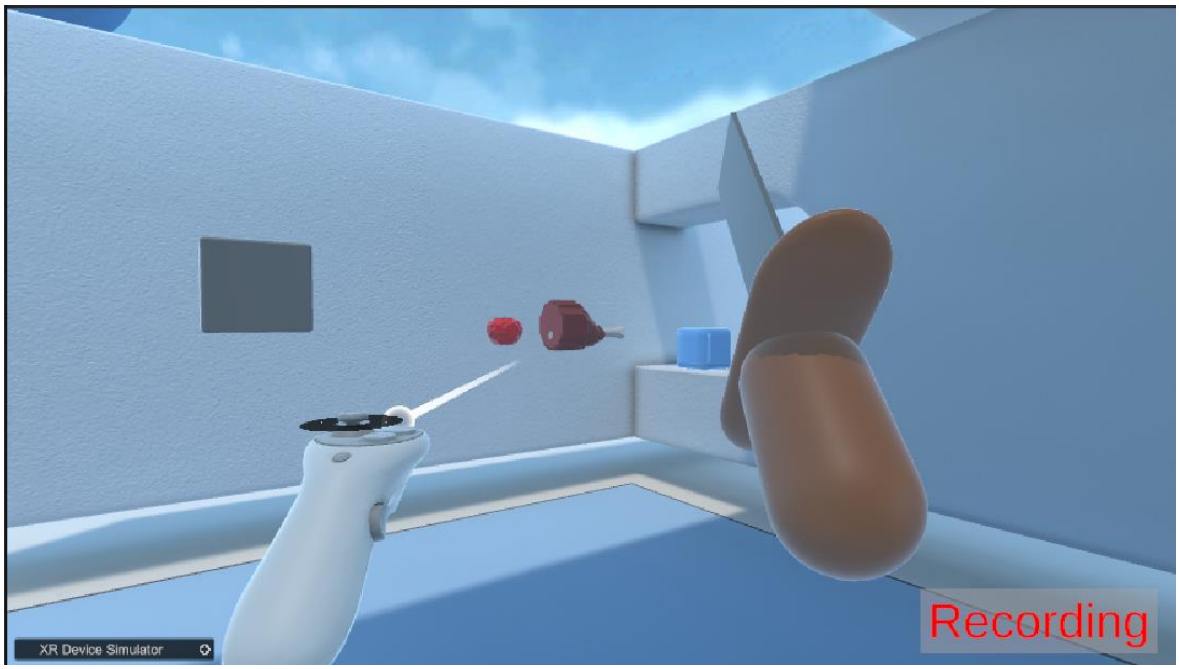
3.2.3. Praćenje staze sječiva

U aplikaciji se nalazi skripta „TransformDataTracker.cs“ koja je zadužena za praćenje „Transform“ komponente objekta na kojem se nalazi. Komponenta sprema poziciju i rotaciju u polje s oznakom trenutka u kojem su logirani. Podatci se mogu spremati relativno u odnosu na neki objekt (na primjer možemo definirati da se podatci o alatu za rezanje spremaju u odnosu na najbliži objekt za rezanje). Ako se ne definira objekt za relativno praćenje podataka, onda se koristi pozicija i rotacija objekta koji pratimo u trenutku pokretanja snimanja. Skripta „TrackRightController.cs“ je zadužena za upravljanje `TransformDataTracker` na desnoj igraćoj palici, odnosno na alatu za rezanje objekta. `TrackRightController` pokreće i završava snimanje nakon

primitka naredbe korisnika pritiskom na gumb na igraćoj palici (Kod 3.2). Kada je pokrenuto snimanje na ekranu se nalazi tekst „Recording“ (Slika 3.3)

```
void Start()
    {
        List<InputDevice> rightHandDevices = new();

UnityEngine.XR.InputDevices.GetDevicesAtXRNode(UnityEngine.XR
.XRNode.RightHand, rightHandDevices);

        if(rightHandDevices.Count == 1)
            device = rightHandDevices[0];
    }

    void Update()
    {
        bool primaryValue;
        if (device != null &&
device.TryGetFeatureValue(UnityEngine.XR.CommonUsages.primary
Button, out primaryValue) && primaryValue)
        {
            Debug.Log("Primary button is pressed.");
            if (tracker.isCollectingData) EndRecording();
            else StartRecording();
        }
        if (PrimaryButtonHolder)
        {
            if (tracker.isCollectingData)
            {
                EndRecording();
            }
            else StartRecording();
            PrimaryButtonHolder = false;
        }
    }
```

Kod 3.2 – Skripta za upravljanje praćenjem podataka



Slika 3.3 - Pokrenuto praćenje alata za rezanje objekata

3.2.4. Izvoz podataka

Za izvoz podataka zaduženja je skripta „TrackRightController.cs“ koja sprema podatke nakon završavanja svakog snimanja. Podatci se izvoze u CSV formatu gdje su stupci redom: redni broj snimanja, vrijeme nakon započetog snimanja, pozicija (x, y, z komponenta) i rotacija (quaternion).

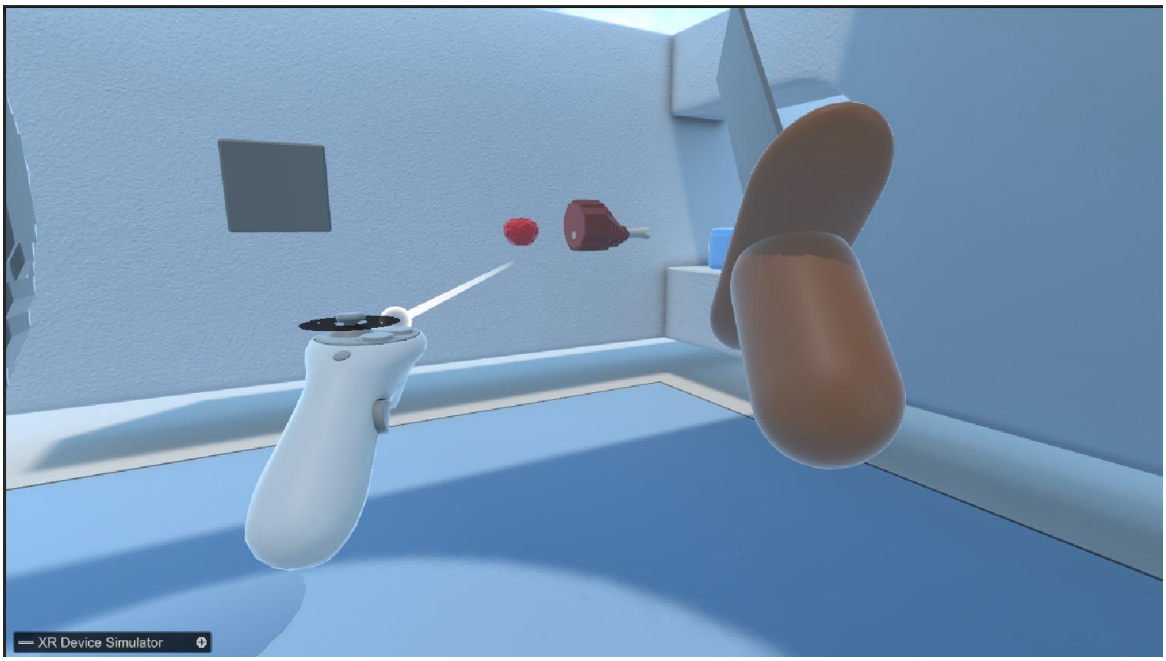
```
public void ExportData()
    {
        string[] headers = { "recordingNumber", "timestamp",
"position", "rotation" };
        List<string[]> rows = new List<string[]>();

        rows.Add(headers);

        int i = 0;

        foreach(TrackedData[] list in trackingDataLists)
        {
            foreach (TrackedData d in list)
            {
```

```
                rows.Add(new string[] { i.ToString(),
d.timestamp.ToString(), d.position.ToString(),
d.rotation.ToString() });
            }

            i++;
        }

        // Generate CSV string
        StringBuilder sb = new StringBuilder();
        foreach (string[] row in rows)
        {
            sb.AppendLine(string.Join(",", row));
        }

        // Save CSV data to a file
        string filePath = Application.dataPath + "/data.csv";
        File.WriteAllText(filePath, sb.ToString());
    }
```

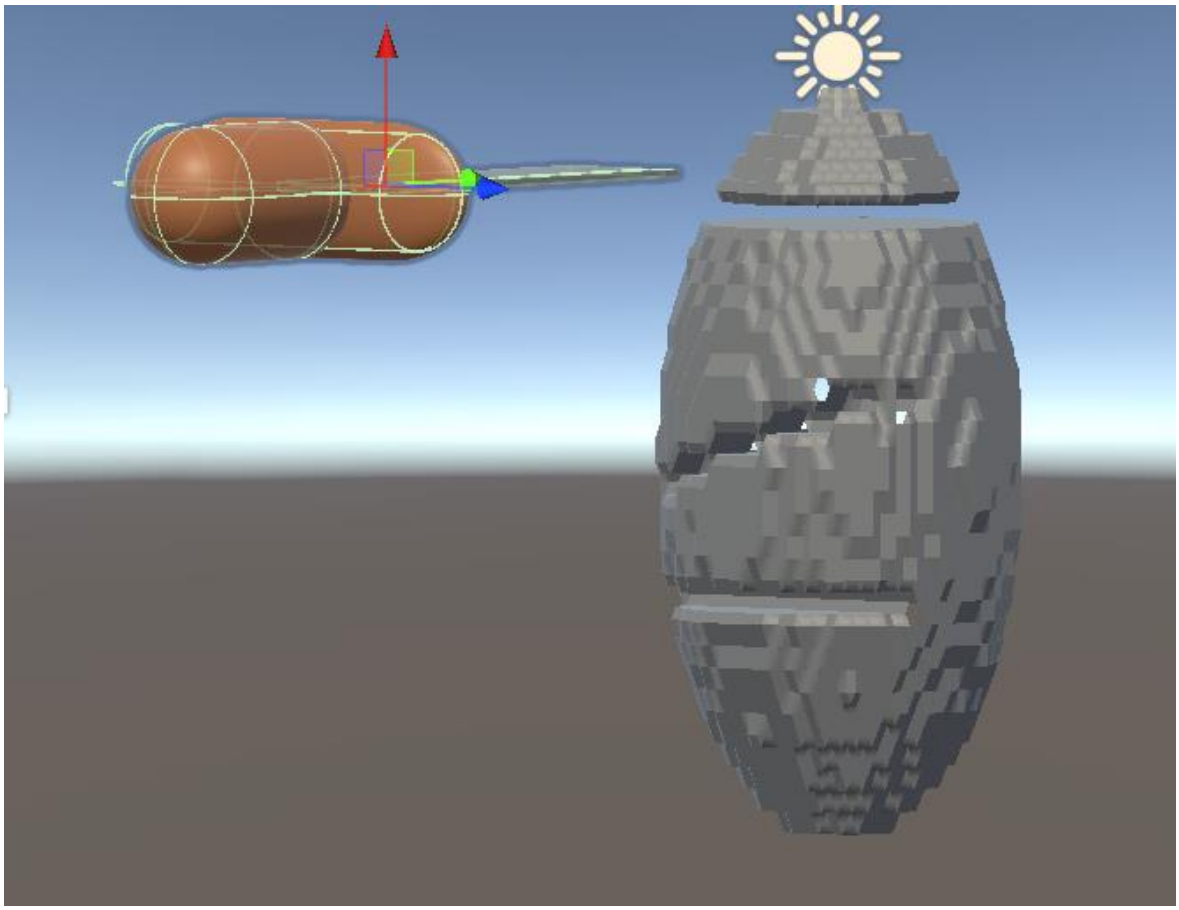
Kod 3.3 – Izvoz podataka o praćenju alata za rezanje

3.3. Upute za korištenje

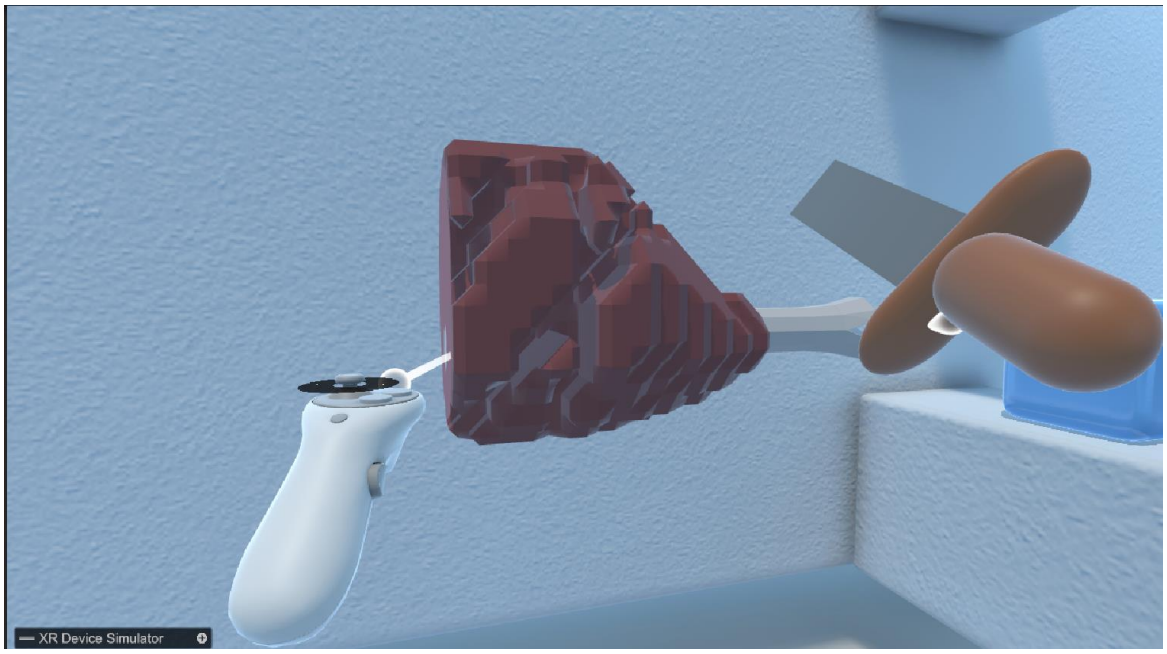
Pokretanjem aplikacije korisniku se prikazuje scena kao na slici (Slika 3.4). Korisnik se može kretati u prostoru korištenjem lijeve igraće palice. Pritiskom na „primary“ gumb desne palice se započinje/završava snimanje pokreta desne palice. Korištenjem desne palice korisnik u prostoru pomiče i rotira alat za rezanje objekta. Na slikama Slika 3.6 i Slika 3.7 se nalaze primjeri rezultata rezanja objekata u VR aplikaciji.



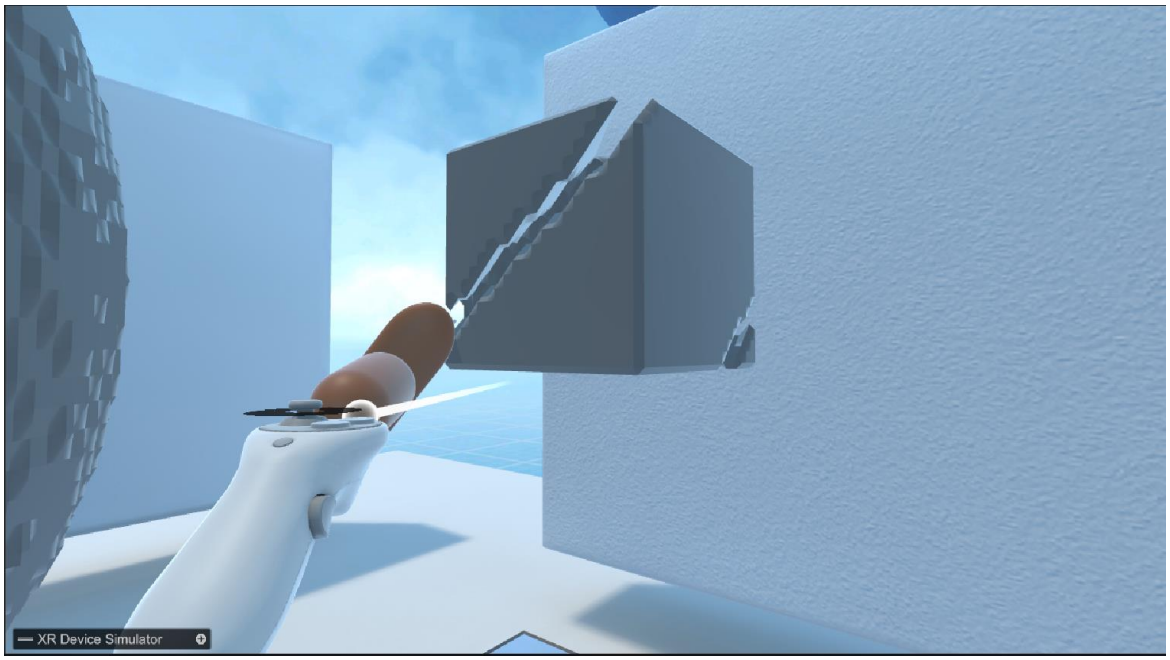
Slika 3.4 - Početni ekran aplikacije



Slika 3.5 - Potpuno i djelomično zarezani objekt



Slika 3.6 - Primjer rezanja objekta 1



Slika 3.7 - Primjer rezanja objekta 2

3.4. Potencijalne nadogradnje sustava

Pristup korištenja voxela i MC algoritama ima prednost jednostavnog rezanja objekta, ali postoje brojni nedostatci. Glavni nedostatci su memorijsko zauzeće za velike objekte ili

veliku preciznost (mali razmak točaka). Ako smanjimo udaljenost točaka kako bi smanjili memorijsko zauzeće gubi se oblik/detalji originalnog objekta. Problem je i visoka složenost MC algoritma koji je u originalnom obliku $O(n^3)$ vremenske složenosti. Postoje implementacije niže složenosti, npr. većina jediničnih kocaka je potpuno unutar ili izvan pa njih ne moramo prolaziti [1]. Bolji pristup problemu bi bio da se definiraju dvije poluravnine koje kreću od istog pravca i definiraju oblik oštrice i onda preko tih poluravnina dobiti koje poligone na objektu treba izbaciti i dodati nove koji ovise o poziciji i rotaciji sječiva (poluravnina koje ju definiraju). Razdvajanje trokuta moglo bi se raditi preko određivanja baricentričnih koordinata presjeka ravnina i poligona objekta [2]. U radu "A Cutting Remesh Method Based on Barycentric Coordinates for 2D Triangulation Mesh," (T. Shijie, Z. Jinjin, Z. Hongjun 2018.) autori iznose rješenje za razdvajanje mreže poligona za neku krivulju, ali u 2D sustavu. Za implementaciju u aplikaciju potrebno je razraditi rješenje u 3D.

Drugi veći problem u implementaciji je ovisnost o tome da objekt ima „MeshCollider“ jer Unity ne dozvoljava da su objekti koji sadrže „MeshCollider“ dinamički (da utječe fizika simulacije na objekt) i automatski postavlja zastavicu IsKinematic.

Zanimljivi dodatak aplikaciji bio bi taktilna povratna informacije putem haptičkih motora u igraćim palicama [2].

Također aplikacija bi trebala nuditi više formata izvoza podataka.

Zaključak

Dinamičko ažuriranje i interakcija s virtualnim objektima definiranim poligonima je složena tema kojoj se može pristupiti na puno načina. U ovom radu se to izvodi operacijama nad 3D poljem točaka koje reprezentiraju sadrži li ih objekt ili ne. Objekte koje želimo rezati pretvaramo u to 3D polje praćenjem zrake i sudarima s mrežom trokuta objekta. To 3D polje se zatim iscrtava kao novi objekt korištenjem Marching Cubes algoritma. Takav postupak nam omogućuje da proizvoljno mijenjamo podatke u polju i ponovno pozivamo MC algoritam da nam iscrta ažurirani objekt. Tijekom rezanja pratimo podatke o poziciji i rotaciji sječiva koji se onda mogu spremiti i koristiti za treniranje modela strojnog učenja za učenje rezanja robotskom rukom. Aplikacija je implementirana u VR okruženju te se sječivo nalazi na poziciji desne igraće palice.

Literatura

[1] Q. Du, J. Zhao, L. Shi and L. Wang, "Efficient improved marching cubes algorithm," Proceedings of 2012 2nd International Conference on Computer Science and Network Technology, Changchun, China, 2012, pp. 416-419, doi: 10.1109/ICCSNT.2012.6525967. keywords: {3D reconstruction;marching cubes;surface rendering;visualization tool kit (VTK)}

[2] J. Williams, G. T. O'Neill and Won-Sook Lee, "Interactive 3D haptic carving using combined voxels and mesh," 2008 IEEE International Workshop on Haptic Audio visual Environments and Games, Ottawa, ON, Canada, 2008, pp. 108-113, doi: 10.1109/HAVE.2008.4685308. keywords: {Haptic interfaces;Animation;Real time systems;Medical simulation;Shape;Solids;Conferences;Medical services;Biomedical equipment;Visualization;carving;haptic;interactive;3D;computer animation}

[3] T. Shijie, Z. Jinjin and Z. Hongjun, "A Cutting Remesh Method Based on Barycentric Coordinates for 2D Triangulation Mesh," 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), Xi'an, China, 2018, pp. 1252-1255, doi: 10.1109/IMCEC.2018.8469652. keywords: {Trajectory;IP networks;Computational modeling;Topology;Deformable models;Solid modeling;Cutting tools;remesh;duplication point;barycentric coordinate;2D cutting}