SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6239

VIZUALIZACIJA STATISTIČKIH PODATAKA UPOTREBOM HTC VIVE UREĐAJA

Adrian Komadina

Zagreb, lipanj 2019.

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

Zagreb, 13. ožujka 2019.

ZAVRŠNI ZADATAK br. 6239

Pristupnik: **Adrian Komadina (0036499396)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Vizualizacija statističkih podataka upotrebom HTC Vive uređaja**

Opis zadatka:

Proučiti različite načine prikaza statističkih podataka, posebice obratiti pažnju na mogućnosti prikaza u 3D prostoru. Proučiti mogućnosti uređaja HTC Vive za prikaz i interakciju s objektima u sceni. Razraditi prikaz scene u kojoj se nalaze prikazani statistički podaci u obliku različitih dijagrama. Načiniti aplikaciju koja će omogućiti prikaz scene uz korištenje uređaja HTC Vive te interakciju s pojedinim elementima scene. Razmotriti i diskutirati moguće probleme. Načiniti ocjenu ostvarenih rezultata.
Izraditi odgovarajući programski proizvod, koristiti grafički programski pogon Unity i HTC Vive uređaj. Rezultate rada načiniti dostupne putem Interneta. Radu priložiti algoritme, izvorne kodove i rezultate uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

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

Mentor:
Prof. dr. sc. Željka Mihajlović

Predsjednik odbora za završni rad modula:
Doc. dr. sc. Marko Čupić

Dielovođa:
Izv. prof. dr. sc. Tomislav Hrkać

Sadržaj

1. Uvod

Iz godine u godinu svjedočimo ubrzanom rastu količine znanja, pa tako i podataka kojima ljudsko društvo raspolaže. Shodno tome zadatak računalne obrade svih tih podataka je postao izuzetno zahtjevan posao, međutim tu rad s našim podacima ne staje, potrebno je i sve te podatke vizualno prikazati na primjeren način kako bismo uspjeli izvući i razumjeti bitne podatke. Sljedeći problem leži u tome što kompleksnije podatke najčešće moramo prikazati u više od dvije dimenzije, a naša je tehnologija primarno razvijena za 2D ekrane. Čak i prikazom podataka u tri dimenzije na 2D ekranima nećemo moći dobiti prikaz kakav stvarno želimo zbog nepostojanja dubine. Također interakcija s takvim objektima je prilično ograničena, stoga sve upućuje na to da se istraži nov način za prikaz takvih podataka.

Unazad par godina razvila se tehnologija koja nam upravo omogućava prikaz i sudjelovanje u virtualnom 3D prostoru odnosno virtualnoj stvarnosti. Iako je njen razvoj tek na početku već se može naslutiti njena korist u mnogim područjima pa tako i u prikazu statističkih podataka.

Ovaj će rad na tragu toga biti usredotočen na stvaranje različitih načina prikaza statističkih podataka u virtualnom okruženju, te interakcije s njima. Nakon uvoda i upoznavanja s ključnim tehnologijama korištenim u ovome radu upoznat ćemo se sa postupkom stvaranja svake od reprezentacije podataka. Zatim ćemo pogledati kako postaviti scenu u virtualno okruženje i uparit je s uređajem HTC Vive. Nakon toga ćemo opisati postupak interakcije s pojedinim elementima scene i na kraju razmotriti kakav veliki skupovi podataka imaju utjecaj na izvođenje našeg programa.

U svrhu toga korišteni su uređaj za prikaz virtualne stvarnosti HTC Vive i Unity grafički pogon za izradu aplikacije. Nešto više o navedenim tehnologija reći ćemo u idućem poglavlju.

2. Korištene tehnologije

2.1 Uređaj HTC Vive

HTC Vive je uređaj za prikaz virtualne stvarnosti razvijen od tvrtki HTC i Valve Corporation. Virtualnu stvarnost najjednostavnije možemo opisati kao oblik interaktivne računalne simulacije prilikom kojega se sudionik osjeća kao pripadnik umjetno stvorenog okruženja.

Prvi je put predstavljen 2015. godine, a već je iduće godine postao dostupan za potrošače.

Slika 2.1. – Dijelovi HTC Vive uređaja

HTC Vive u svome osnovnom pakiranju donosi: ureaj za prikaz virtualne stvarnosti odnosno naočale, dva kontrolera i dvije bazne stanice (Slika 2.1.).

Naočale za virtualnu stvarnost za prikaz slike koriste dva OLED ekrana (za svako oko po jedan), svaki rezolucije 1080x1200 piksela. Ekran ima veliku brzinu osvježavanja od 90Hz koja nam pomaže u svladavanju osjećaja mučnine pri kretanja u virtualnom prostoru te kut gledanja od 110 stupnjeva. Kako bi dobili

percepciju dubine objekata odnosno 3D prostora na ravnom 2D ekranu potrebno je koristiti leće u naočalama. S obzirom da zrake svijetla upadaju na leću pod različitim kutovima percipiramo sliku udaljenije nego što ona zapravo jest. Konkretne leće koje se koriste u ovome uređaju su Fresnelove leće (Slika 2.2.) što zapravo znači da su složene od više manjih dijelova. Broj takvih dijelova utječe negativno na maksimalnu svjetlinu koja dolazi do naših očiju, međutim bolje utječe na razlučivost slike. Takvim se lećama postiže da uređaj bude tanji i lakši.

Slika 2.2. – Fresnelove leće

Iz sigurnosnih se razloga na prednjoj strani nalazi kamera s kojom imamo mogućnost promatranja okoline što nam u praksi donosi mogućnost da znamo gdje se nalaze rubovi prostorije u kojoj je smješten naš HTC Vive. Na uređaju se još nalazi i nekoliko infracrvenih senzora koji preko komunikacije s baznim stanicama registriraju korisnikov položaj glave u prostoru. Pored toga uređaj ima još i neke dodatne senzore kao što su žiroskop i akcelerometar za praćenje kretnji.

Bazne stanice emitiraju infracrvene impulse 60 puta u sekundi i na taj način detektiraju položaj glavnog uređaja i kontrolera. Također one stavljene dijagonalno u prostoru zajedno kreiraju virtualno okruženje od maksimalno 4.5x4.5 metara površine.

Kontroleri sadrže nekoliko tipki koje služe za upravljanje u virtualnom okruženju, te svaki sadrži preko 20 infracrvenih senzora koji se nalaze na prstenu kontrolera i komuniciraju s baznim stanicama i omogućavaju precizno praćenje.

Za razliku od drugih konkurentnih uređaja kao što su Oculus Rift, Samsung Gear VR, PlayStation VR i drugi, HTC Vive od prvog dana pruža podršku kretanja kroz virtualni prostor, a ne samo njegovo promatranje, pri čemu upravo koristi brojne već opisane infracrvene senzora i dvije bazne stanice.

Nedugo nakon predstavljanja osnovnog proizvoda uveden je i novi dodatak HTC Vive tracker. Taj je uređaj sastavljen od mnogo infracrvenih senzora, te nam služi za praćenje nekog posebnog dijela tijela koji želimo pratiti pa se upotrebom više takvih dodataka otvaraju nove mogućnosti u svijetu virtualne stvarnosti. 2018. godine u prodaju je izašla nova verzija uređaja nazvana HTC Vive Pro koja nosi brojna unaprjeđenja u odnosu na osnovni proizvod.

2.2 Pogon za izradu igara Unity

Kao podlogu za stvaranje svih potrebnih objekata koje ćemo koristiti u prikazu statističkih podataka, njihovih ponašanja, te same scene koja okružuje igrača u virtualnom trodimenzionalnom prostoru koristit ćemo Unity3D pogon za izradu igara (eng. Game engine). Pogon za izradu igara je softversko-razvojno okruženje koje pruža razne funkcionalnosti. Neke od najvažnijih su: pogon prikazivanje 2D i 3D grafike, otkrivanje sudara, skriptiranje, animacije, zvuk, umjetna inteligencija i mnoge druge. Unity je prvi put predstavljen i izdan 2005. godine, te u početku bi podržan jedino na Mac operacijskom sustavu. Danas, 14 godina poslije, Unity je podržan na više od 25 različitih platformi kao što su Windows, Android, IOS, Linux, Steam VR itd. Stoga ga danas s pravom nazivamo višeplatformskim pogonom. Velika je prednost Unity-a njegova besplatna uporaba, jednostavnost uporabe i velika raširenost u svijetu video igara. S obzirom na svoju popularnost Unity nudi skladište imovinom (eng. Asset Store) gdje se mogu naći brojni dodaci napravljeni od drugih ljudi diljem svijeta kako bi nam olakšali izradu svojih radova. Takvi dodaci

mogu uključivati od malih skripta, modela i animacija pa sve do polugotovih dijelova video igara.

Unity za svoje glavno aplikacijsko programsko sučelje (API) pri pisanju skripti korisiti programski jezik C#. Pisanjem skripti omogućit ćemo postojanje interakcija i sveukupne dinamiku u našoj video igri.

Trenutna stabilna verzija Unity pogana izdana 8. travnja 2019. godine nosi naziv Unity 2018.3.12. koju ćemo koristi u izradi ovoga rada.

2.2.1 Skladište imovinom Steam VR

Kao jedan od važnih dodataka za ovaj rad može se pronaći u skladištu imovinom pod nazivom Steam VR. Steam VR je priključnica (eng. Plugin) za Unity što znači da proširuje funkcionalnost osnovnog programa. Ova nam priključnica služi kao most između Unity programa i HTC Vive uređaja kako bismo mogli izvoditi svoj program na tom uređaju. Osim te osnovne funkcije posjeduje još tri važne funkcionalnosti: učitavanje 3D modela za prikaz pozicije kontrolera u sceni, prepoznavanje pritiska tipki na kontroleru i njihovo pridruživanje određenom događaju i pretpostavka položaja korisnikove ruke pri korištenju kontrolera.

3. Modeli prikaza podataka

U ovom ćemo poglavlju opisati izradu četiri vrste grafikona: linijski, stupčasti, trakasti i raspršeni grafikon i karte Europe s prikazom podataka. Opis izrade će se sastojati od opisa sastavnih komponenti koje čine model i opisa skripte koja modelira ponašanje komponenti.

3.1 Obrada podataka

Prije nego što krenemo graditi grafikon trebamo reći nešto o skupu podataka koje ćemo koristiti, jer ipak su oni sastavni dio svakog statičkog postupka. Kao datoteku za čuvanje naših podataka odabrali smo datoteku tipa .csv jer se ona najčešće koristi za čuvanje statističkih podataka. Najčešće takva datoteka u prvom retku sadrži nazive mjerenih varijabli odvojenih zarezom, a u ostalim redcima sadrži vrijednosti opet odvojene zarezom tako da svaka pozicija odgovara svojoj varijabli. Možemo odmah zaključiti kako je ovakav zapis vrlo pogodan za različite vrste obrada.

Ipak za stupčasti i trakasti grafikon koristit ćemo nešto drugačiji zapis. S obzirom da takvi grafovi trebaju sadržavati ime kategorije, ime mjerene vrijednosti i vrijednosti. Za takve grafikone ćemo imati sljedeći zapis: prvi stupac će označavati pojedinu kategoriju, a ostali stupci će izgledati isto kao što je prethodno opisano. Podvrstu tog zapisa upotrijebit ćemo za prikazivanje mape Europe sa željenim podacima. U takvom zapisu će nam kategorije biti imena država na hrvatskom jeziku kako bi ih mogli uspješno prikazati na karti.

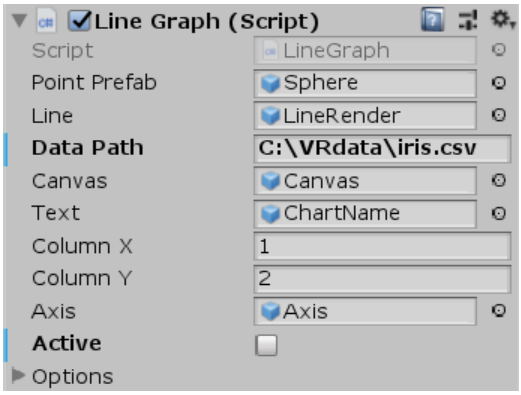
U svrhu obrade jedne takve .csv datoteke stvorili smo skriptu MyCSVReader koja sadrži nekoliko metoda za dohvat podataka iz datoteke. Pa tako imamo metodu ReadPairs koja prima indekse dvaju stupaca iz kojih želimo dohvatiti podatke i put do datoteke, a vraća sortiran rječnik sa parovima vrijednosti ovisno o stupcima koje smo odredili. Ovakav dohvat podataka nam je pogodan za modeliranje linijskog grafikona. Za ostale grafikone ćemo imati drugačije metode ovisno o prirodi grafikona. Pa ćemo tako za stupčasti i trakasti grafikon koristiti metodu

RadDataCategory koja vraća klasu categoryData sa listom imena kategorija i listom vrijednosti za odabrani stupac. Za raspršeni grafikon koristit ćemo metodu ReadData koja na osnovi tri zadana stupca vraća listu lista podataka za svaki navedeni stupac. Također postoji i dodatna metoda AllColumns koja vraća listu naziva svih stupaca i ReadColumns samo za određene stupce.

3.2 Linijski grafikon

Modeliranje svih grafikona pa tako i linijskog izvest ćemo dinamički, što znači da će naš grafikon predstavljati jedna skripta, u ovom slučaju LineGraph.cs koja će se ovisno o danim globalnim parametrima pobrinuti za pravilo iscrtavanje grafikona u 3D prostoru. Kako bi skripta za to bila sposobna, mora sadržavati referencu na nekoliko predložaka objekata (Prefab) koji su nam potrebni za izgradnju, a to su: kugla (točke grafikona), linija koja povezuje točke, ravna podloga, tekst i izduženi valjak (osi grafikona). Predlošci objekata nam služe kako bi cjelovito spremili bilo koji objekt koji smo prije napravili da bi ga mogli koristiti instanciranjem negdje u budućnosti. U Unity-u se ovakvi objekti jednostavno predaju skripti preko javnih globalnih varijabli. Unity nam omogućuje da početne vrijednosti tih varijabli zadamo u uređivaču (Slika 3.1.). Osim predložaka objekata kao javne globalne varijable zadat ćemo i put do datoteke, stupce koje učitavamo, varijablu koja određuje aktivnost grafikona i popis svih dostupnih varijabli zbog toga što želimo dati mogućnost korisniku i drugim skriptama na njihovo korištenje.

```
public GameObject pointPrefab;
public GameObject line;
public string dataPath;
public GameObject canvas;
public GameObject text;
public int columnX = 1;
public int columnY = 2;
public GameObject axis;
public bool active=false;
public List<string> options;
```



Slika 3.1. – Veza javnih globalnih varijabli u skripti i uređivaču

Svaka skripta u Unity-u mora sadržavati predefiniranu metodu Start() koja se poziva pri stvaranju objekta. U toj ćemo metodi zapamtiti početni položaj, rotaciju i veličinu koji će nam kasnije zatrebati i učitati određeni skup podataka. Druga predefinirana metoda svake skripte je Update() koja se aktivira prilikom svakog novog iscrtavanja slike na ekranu. U njoj ćemo provjeravati jeli u međuvremenu korisnik promijenio veličinu ili položaj grafikona te ćemo na temelju tog saznanja obaviti potrebne radnje.

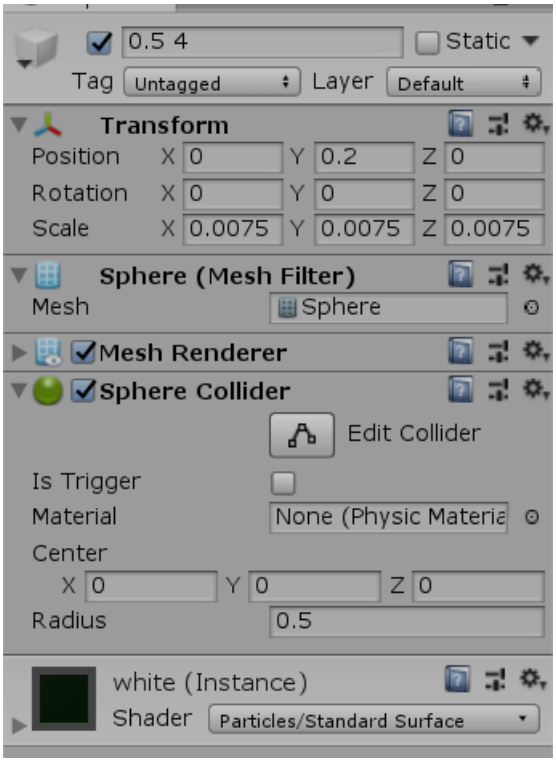
Središnja metoda ove skripte bit će drawGraph() čiji je zadatak pravilo iscrtati grafikon na temelju učitanih podataka. Sve što ona zapravo radi je stvara konkretne objekte iz predložaka i mijenja im svojstva kao što su pozicija, veličina, rotacija i boja. Za primjer ćemo uzeti dio koda za iscrtavanje točaka grafikona (Slika 3.2.).

```
//Drawing Points
Vector3 startPoint=new Vector3(0,0,0);
Vector3 endPoint=new Vector3(0,0,0);
int i=0;
List<Vector3> positions=new List<Vector3>();
List<GameObject> points=new List<GameObject>();
foreach(float x in keys){

    float posX=( (x - xMin) / (xMax - xMin) )*scale;
    float posY=( (data[x] - yMin) / (yMax - yMin) )*scale;

    string dataPointName=x +" "+ data[x];

    GameObject pointInst=Instantiate(pointPrefab);
    Vector3 position=new Vector3(posX,posY,0);
    pointInst.transform.localPosition=position;
    positions.Add(position);

    pointInst.transform.SetParent(transform, false);
    pointInst.GetComponent<Renderer>().material.color = new Color(posX,posY,0.0f, 1.0f);
    pointInst.transform.name=dataPointName;
    float scaleFactor=0.3f/40;
    pointInst.transform.localScale=new Vector3(scale*scaleFactor,scale*scaleFactor,scale*scaleFactor);
    points.Add(pointInst);
}
pointList=points;
pointPositions=positions;
drawLines();
```

Slika 3.2. – Isječak koda za iscrtavanje točaka

Točke se iscrtavaju tako da iteriramo po rječniku koji sadrži parove vrijednost na x osi i vrijednost na y osi. Te vrijednosti najprije normiramo na interval [-1 , 1] kako bi imali ograničenje veličine grafikona, a pritom sačuvali originalne odnose između točaka te potom te vrijednosti množimo s odgovarajućim faktorom scale ovisno o

tome koliko velik grafikon želimo dobiti. Tako izračunate vrijednosti predstavljaju pozicije naših točaka u lokalnom koordinatnom sustavu grafikona. Objekt u 3D prostoru stvaramo pozivom metode Instantiate koja nam vraća referencu na stvoreni objekt te ju spremamo u varijablu pointInst tipa GameObject koji predstavlja bilo koji objekt u Unity-u. Sada preko te reference možemo lako mijenjati pojedina svojstva objekta. Pa tako poziciju objekta mijenjamo spremanjem Vector3 varijable u pointInst.transform.position. Sva svojstva objekta možemo također naći i promijeniti direktno u uređivaču (Slika 3.3.).



Slika 3.3. – Svojstva objekta u uređivaču

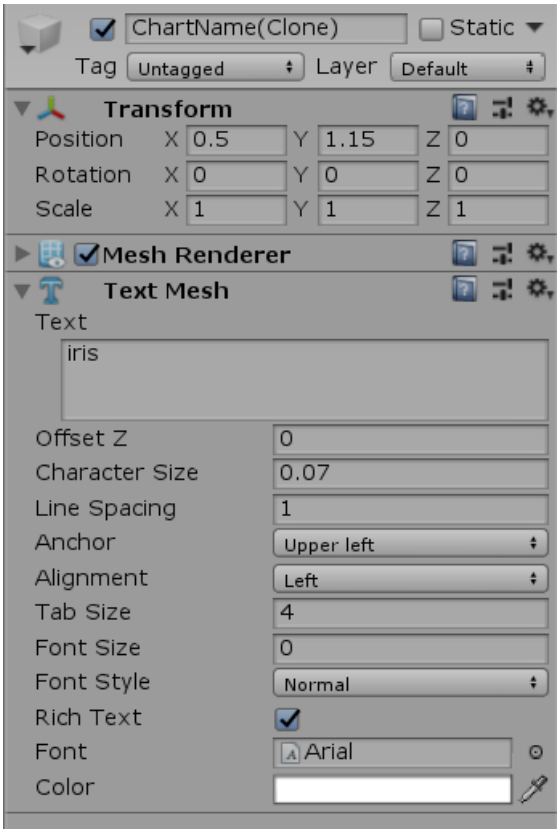
Osim izmjena pozicije, rotacije, veličine, imena i boje želimo još svaku točku vezati za lokalni koordinatni sustav skripte odnosno cijelog grafikona. Na taj smo način osigurali da primjenom neke od transformacija na cijeli grafikon ostane sačuvan originalni uređaj unutar njega odnosno da se ista transformacija primjeni i na sve njegove vezane elemente. To smo ostvarili metodom Setparent nad svojstvom transform svakog elementa. Točke također trebamo u procesu spremati u listu kako bi znali kasnije iscrtati linije između svake od njih. Proces iscrtavanja linija ostavljamo metodi drawLines() koja se idejno isto izvršava samo uporabom line komponente. Metoda drawGraph() će još obaviti sljedeće radnje: Iscrtavanje

pozadine, osi i imena grafa te pridjeljivanje imena i vrijednosti osima. Sve se te radnje odvijaju slično kao već opisano iscrtavanje točaka pa ćemo još samo prikazati dio koda odgovoran za iscrtavanje imena grafikona (Slika 3.4.). Ta se radnja obavlja tako da stvorimo objekt tipa text, namjestimo ga na odgovarajuću lokaciju i promijenimo vrijednost i veličinu text komponente.

```
//Creating graph name
GameObject nameTextInst=Instantiate(text);
Transform nameT=nameTextInst.transform;
nameT.SetParent(transform, false);
nameT.localPosition=new Vector3(scale/2,scale+1.5f*scale/10,0);
string[] parts=dataPath.Split('\\');
string[] parts2=parts[parts.Length-1].Split('.');
string graphName=parts2[0];
nameTextInst.GetComponent<TextMesh>().text = graphName;
nameTextInst.GetComponent<TextMesh>().characterSize=0.7f*scale/10;
```

Slika 3.4. – Isječak koda za iscrtavanje imena grafikona

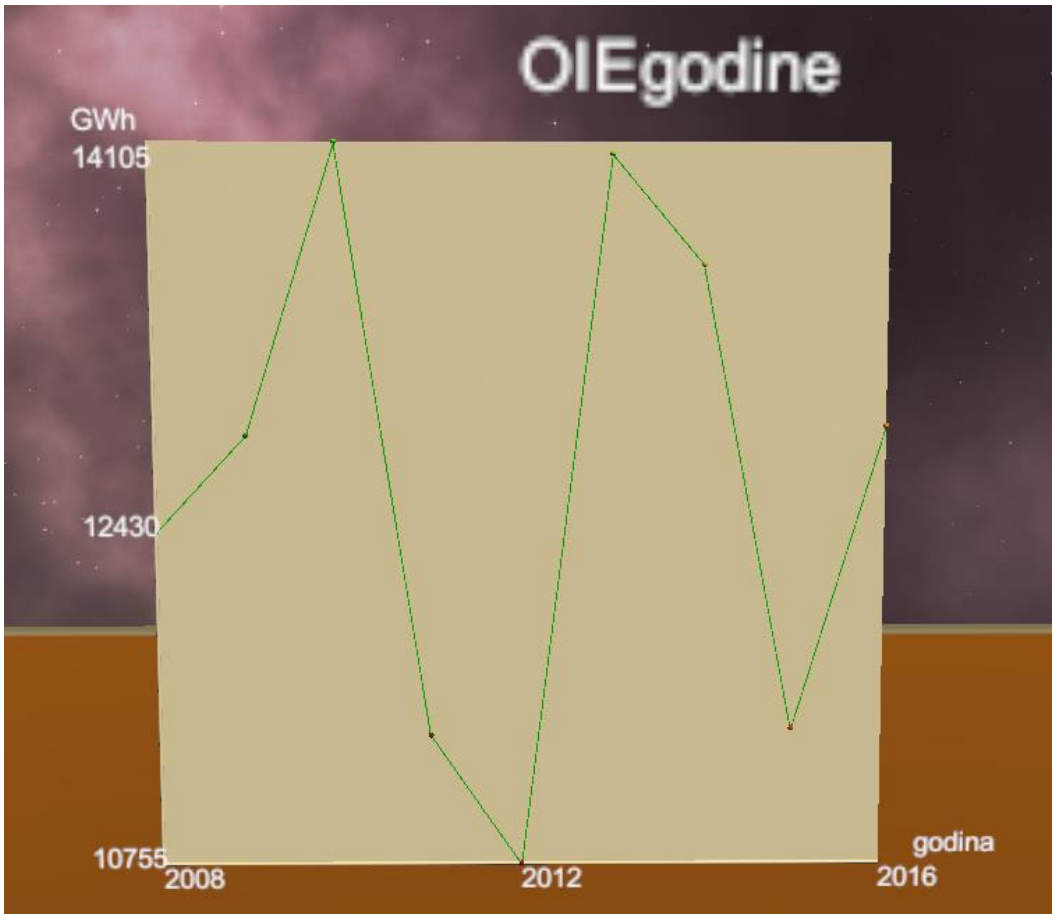
Rezultat stvaranja objekta možemo lako provjeriti u uređivaču (Slika 3.5.)



Slika 3.5. – Ime grafikona u uređivaču

U skripti se još nalaze metode newSource() koja služi za učitavanje nove datoteke ako korisnik tako odluči.

Za testiranje ispravnosti koristiti ćemo skup podataka OIEgodine.csv koji prikazuje energetsku bilancu električne energije od 2008. do 2016. godine u Hrvatskoj. Odabrat ćemo stupce 1 i 2 koji odgovaraju godini i bilanci mjerenoj u GWh za tu godinu. Konačni rezultat možemo vidjeti na slici 3.6.



Slika 3.6. – Linijski grafikon

3.3 Stupčasti i trakasti grafikon

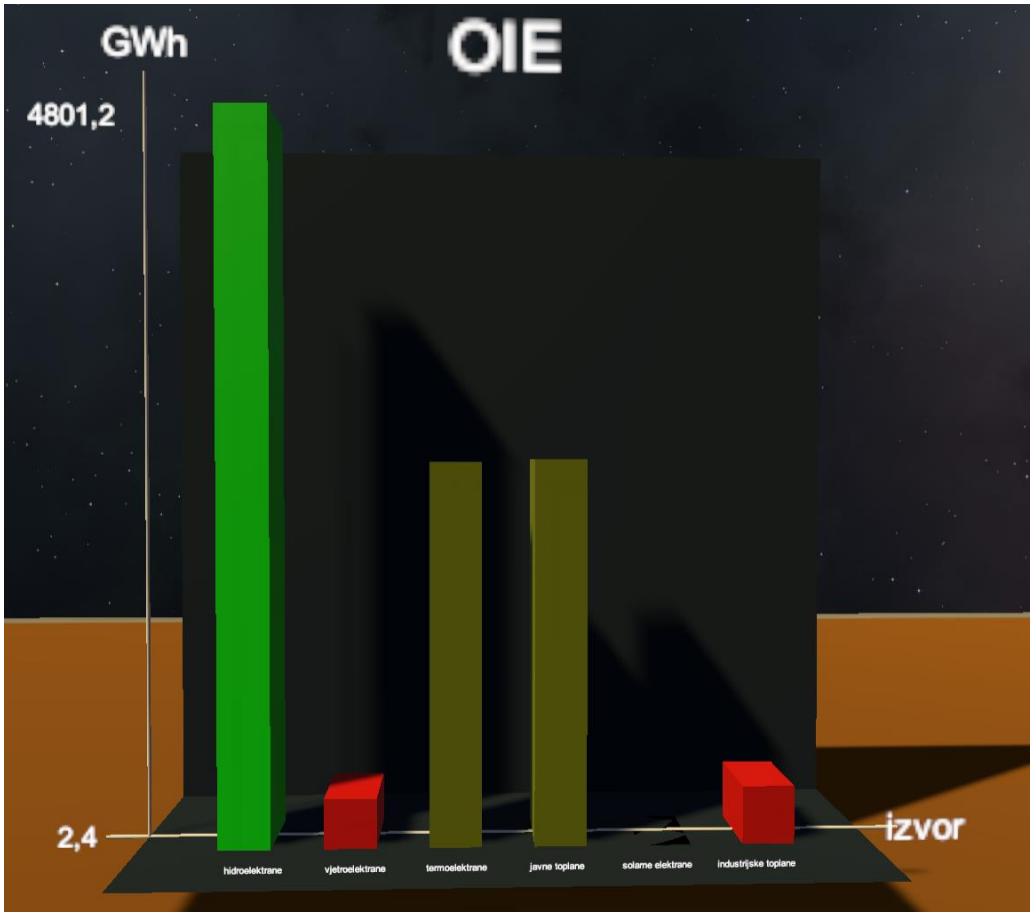
Stvaranje i ponašanje stupčastog grafikona zapisano je u skripti ColumnChart.cs. Za razliku od linijskog grafikona, ovaj se temelji na prikazu podataka u obliku stupaca. Odnosno umjesto sfere predati ćemo skripti kvadar kojem ćemo visinu odrediti temeljem vrijednosti kategorije iz .csv datoteke. Također s obzirom da grafikon namjeravamo koristiti u virtualnom odnosno 3D prostoru dodat ćemo mu dubinu što kod linijskog grafikona ne bi imalo smisla zbog njegove prirode. Kao što smo već napomenuli u poglavlju 3.1 skup podataka koje ova skripta koristi je oblika categoryData koja sadrži listu imena kategorija i listu vrijednosti za odabrani stupac, a podaci se pune pozivom metode ReadDataCategory iz MyCSVReader skripte. Na slici 3.7. prikazan je isječak koda čija je odgovornost iscrtati stupce grafikona ovisno o učitanim podacima.

```
//Iscrtavanje stupaca grafa
for(int i=0;i<dataSet1.Count;++i){
    float y = ((dataSet2[i]) - yMin) / (yMax - yMin);
    GameObject columnInst=Instantiate(column, new Vector3(i, y, 0)*scale, Quaternion.identity);
    Transform columnInstT=columnInst.transform;
    string dataPointName=dataSet1[i] + " "+ dataSet2[i];
    columnInst.GetComponent<Renderer>().material.color = new Color(1-(dataSet2[i]/yMax),dataSet2[i]/yMax,0, 1.0f);
    columnInstT.name=dataPointName;
    columnInstT.SetParent(transform, false);
    columnInstT.localScale=new Vector3(0.5f*(scale/(dataSet1.Count+1)),y*scale,scale/10);
    columnInstT.localPosition=new Vector3((i+1)*(scale/(dataSet1.Count+1)),(y*scale)/2,0);
}
```

Slika 3.7. – Isječak koda za iscrtavanje stupaca

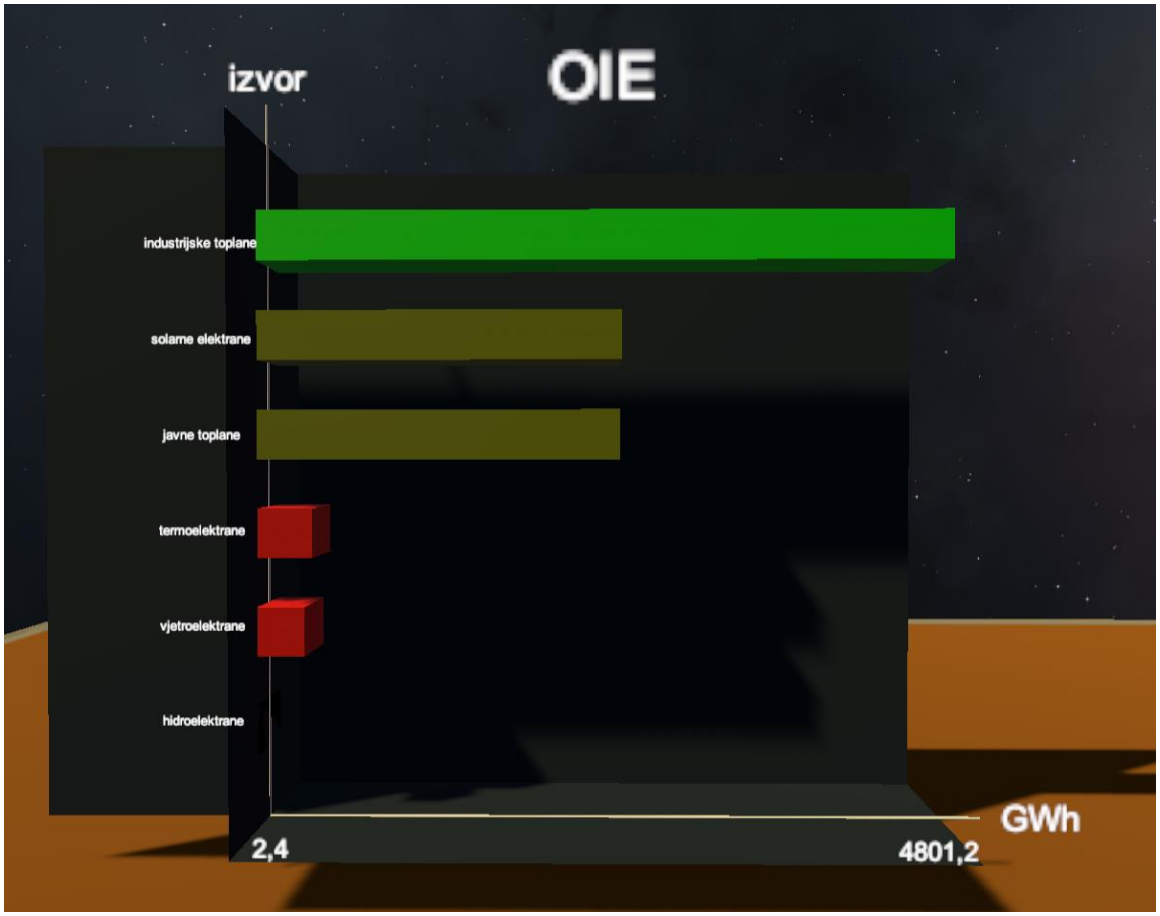
Novosti u odnosu na linijski grafikon su i dodavanje obrubnica grafikona i imena pojedine kategorije ispod svakog stupca.

Skup podataka koje smo koristili za demonstraciju je popis obnovljivih izvora energije i energetsku bilancu električne energije za pojedinu kategoriju izraženu u GWh za 2012. godinu. Kako grafikon u 3D prostoru izgleda vidljivo je na slici 3.8.



Slika 3.8. – Stupčasti grafikon

Izrada trakastog grafikona bit će skoro identična izradi stupčastog. Ono što ćemo promijeniti je to da ćemo prije iscrtavanja stupaca sortirati podatke tako da kategorije s najvećom vrijednosti budu na vrhu, a one s najmanjom na dnu. Pri izradi stupaca ćemo ih samo zarotirati tako da budu horizontalno položeni u odnosu na stupčasti grafikon, a ostali koraci će isto izgledati. Prikaz takvog grafikona u prostoru možemo vidjeti na slici 3.9.



Slika 3.9. – Trakasti grafikon

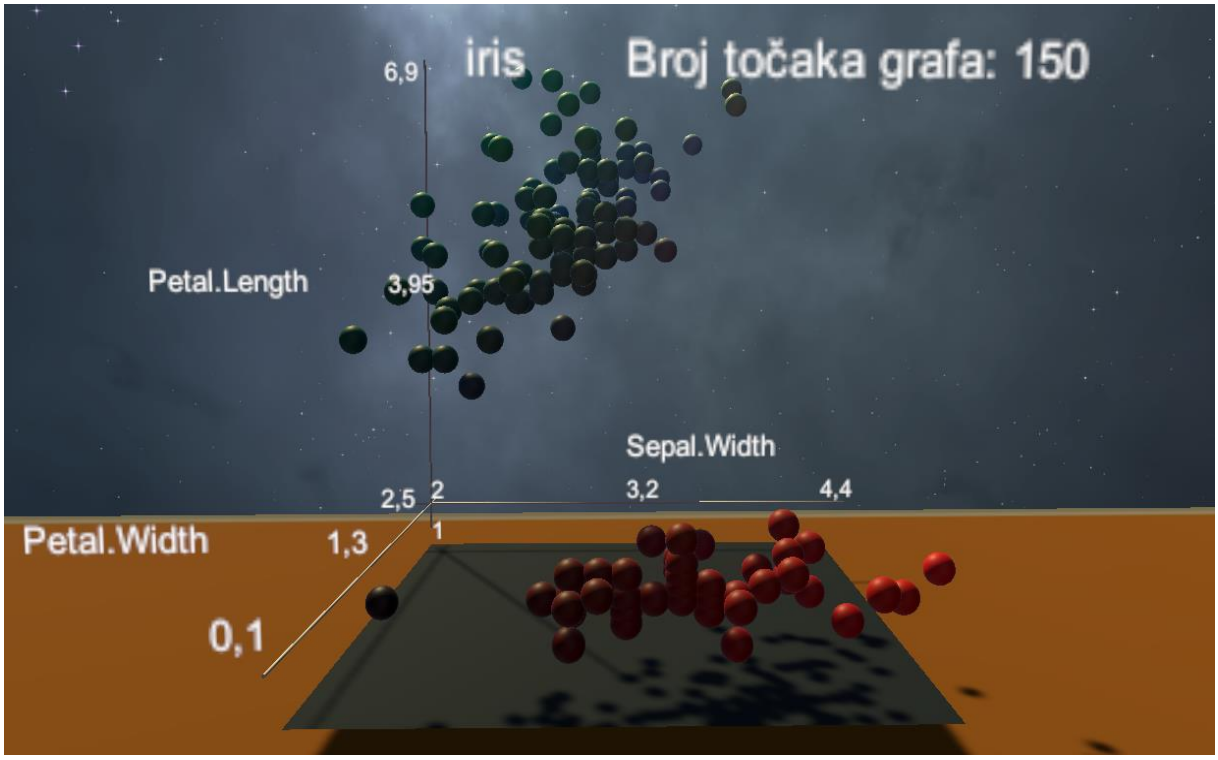
3.4 Raspršeni grafikon

Raspršeni se grafikon temelji na prikazu jediničnih podataka u tri dimenzije, svaka je točka određeni trojcem odabranih varijabli. U skladu s time, ovaj je grafikon zapravo proširenje linijskog za dodatnu dimenziju te je kao takav idealan za prikaz u virtualnom 3D okruženju. Podatke za njegovu izradu dohvaćamo funkcijom ReadData iz myCSVReader skripte koja nam vraća listu koja sadrži u sebi tri liste od kojih svaka predstavlja podatke jedne varijable. Točke grafikona će biti na jednaki način iscrtavane kao što je to opisano kod linijskog grafikona, jedino ćemo uzeti u obzir i z koordinatu na temelju treće liste iz skupa podataka.

Ostali elementi koji čine ovaj grafikon su: baza, osi(x, y, z), vrijednosti na osima (minimum, maksimum, sredina) i naslov. S obzirom da ovakvi skupovi podataka

mogu biti jako veliki zgodno je imati informaciju o broju iscrtanih točaka, pa ćemo i to prikazati pokraj naslova, te ćemo ga koristiti u poglavlju 6. kada budemo razmatrali odnos veličine skupa podataka i performansi našeg programa.

Za demonstraciju ovog grafikona uzeti ćemo skup podataka iris.csv koja sadrži podatke mjerenja raznih karakteristika cvjetova irisa. Odabrati ćemo 3, 4 i 5 stupac koji odgovaraju redom širini lapova, dužini latica i širini latica. Kako grafikon izgleda u 3D prostoru prikazuje slika 3.10.



Slika 3.10. – Raspršeni grafikon

3.5 Karta Europe

Sljedeća je ideja napraviti određeni prikaz podataka ovisno o geografskom području jer je ipak takav prikaz nemoguće prikazati u dvije dimenzije, a vizualno je prilično impresivan i jednostavan korisniku za interpretaciju podataka. Za te potrebe prikazati ćemo kartu Europe i na njoj izmjerenu vrijednost za svaku državu u obliku valjkastih stupaca. Kartu Europe smo napravili tako da smo modelirali jednostavan kvadar razvučen u obliku ploče te sliku Europe preuzetu s Google Maps-a pretvorili u materijal i zalijepili na ploču. Stvaranje stupaca kao i prije izvršavat će se dinamički preko skripte.

Podatke učitavamo metodom ReadDataCategory gdje će u ovome slučaju prvi stupac(kategorija) biti pojedina država, a drugi mjerena veličina. Kako bi mogli znati gdje će se koji stupac iscrtavati postoji metoda fillKoordinate (Slika 3.11.) koja puni globalnu varijablu rječnik koordinate koji sadržava ime države i Point3D strukturu. Point3D struktura nam služi za čuvanje x, y i z koordinate na jednom mjestu za svaku državu.

```
private void fillKoordinate(){
    koordinate.Add("Spanjolska",new Point3D(0.2213f,0,0.3723f));
    koordinate.Add("Portugal",new Point3D(0.2764f,0,0.3851f));
    koordinate.Add("Italija",new Point3D(0.0254f,0,0.3341f));
    koordinate.Add("Njemacka",new Point3D(0.057f,0,0.158f));
    koordinate.Add("Francuska",new Point3D(0.1441f,0,0.2501f));
    koordinate.Add("Grcka",new Point3D(-0.0857f,0,0.3918f));
    koordinate.Add("Ujedinjeno Kraljevstvo",new Point3D(0.1925f,0,0.1247f));
    koordinate.Add("Irska",new Point3D(0.2751f,0,0.1177f));
    koordinate.Add("Slovenija",new Point3D(0.0018f,0,0.2634f));
    koordinate.Add("Hrvatska",new Point3D(-0.0179f,0,0.2726f));
    koordinate.Add("Madarska",new Point3D(-0.0517f,0,0.2457f));
    koordinate.Add("Slovacka",new Point3D(-0.0548f,0,0.2092f));
    koordinate.Add("Ceska",new Point3D(-0.0055f,0,0.186f));
    koordinate.Add("Poljska",new Point3D(-0.0709f,0,0.1335f));
    koordinate.Add("Danska",new Point3D(0.0672f,0,0.0447f));
    koordinate.Add("Norveska",new Point3D(0.0762f,0,-0.0875f));
    koordinate.Add("Svedska",new Point3D(-0.0042f,0,-0.1028f));
    koordinate.Add("Finska",new Point3D(-0.1328f,0,-0.1314f));
    koordinate.Add("Malta",new Point3D(0.006f,0,0.4564f));
    koordinate.Add("Bugarska",new Point3D(-0.1199f,0,0.3344f));
```

Slika 3.11. – Metoda fillKoordinate

Poziciju svake države odredili smo eksperimentalno, odnosno iscrtavajući kartu i jedan stupac na poziciji (0, 0, 0). Taj smo stupac pomicali po ploči i iz uređivača očitavali lokalne koordinate koje će predstavljati pojedinu državu.

Iscrtavanje stupaca provodit će se na temelju povezivanja retka učitanog iz .csv datoteke i lokacije te države u rječniku koordinata. Na tome mjestu će se iscrtati stupac veličine ovisno o vrijednosti izmjerene veličine. Valja obratiti pozornost na to da imena država u prvom stupcu datoteke moraju odgovarati imenima kako smo ih zadani u metodi fillKordinate.

Kao skup podataka za prikaz funkcionalnosti uzeti ćemo datoteku zivotEU.csv u kojoj se nalaze mjerenja očekivanog životnog vijeka žena i muškaraca u određenoj državi. Kartu Europe na temelju očekivanog životnog vijeka muškaraca možemo vidjeti na slici 3.12.

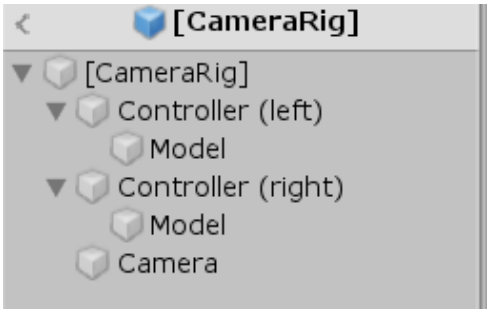
Slika 3.12. – Karta Europe

4. Scena u virtualnoj stvarnosti

Sada kada imamo izrađene prikaze podataka, slijedi izrada virtualne scene u kojoj će oni biti smješteni. Bilo koju 3D scenu izrađenu u Unity-u jednostavno možemo pretvoriti u scenu koja se prikazuje u virtualnoj stvarnosti na način da u Unity-u izaberemo izbornik Edit->Project Settings->Player->XR Settings i tamo označimo podržanost s virtualnom stvarnosti i OpenVR kao alat za virtualnu stvarnost. Na taj ćemo način spajanjem HTC Vive uređaja i pokretanjem aplikacije u uređivaču doći do prikaza scene na našem uređaja za prikaz virtualne stvarnosti. Sljedeće tri glavne značajke koje naša aplikacija treba moći podržati su: kretanje u virtualnoj stvarnosti, izbornici kojima ćemo odabrati prikaze podataka i pokazivač koji će nam služiti za odabir opcija iz izbornika.

4.1 Kretanje po sceni

Nakon što smo omogućili prikaz scene u virtualnoj stvarnosti preuzet ćemo i instalirati dodatak SteamVR. Taj nam dodatak pruža neka olakšanja pri izradi igrača u virtualnoj sceni. Prvi važan objekt kojeg možemo instancirati u našoj sceni je CameraRig (Slika 4.1.). Ubacivanjem ovog objekta postižemo to da u sceni vidimo prikaz dvaju kontrolera koja držimo u 3D prostoru. Sva ta su ponašanja i modeli kontrolera već dani preko uvezene biblioteke, no mi ako želimo sve to možemo u budućnosti promijeniti.

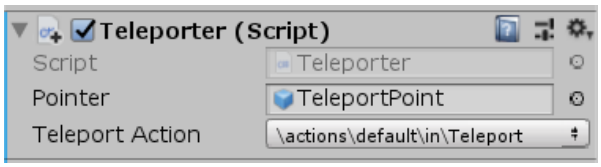


Slika 4.1. – Sadržaj objekta CameraRig

Također kretanje u ovakvome prostoru je podržano odmah kada smo uključili podržanost za virtualnu stvarnost, no naš fizički prostor je u praksi premali za

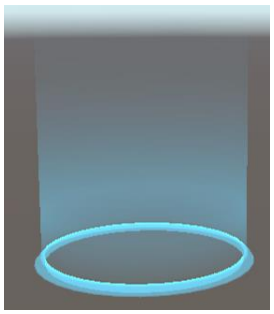
hodanje samo po njemu, a i nepraktično je koristiti običnu šetnju za prelazak velikih udaljenosti. Stoga moramo uvesti određeni sustav teleportacije. Želimo omogućiti korisniku da se pomakom lijevog kontrolera pomiče oznaka za teleportaciju, te pritiskom prema gore na kontroleru se izvršava teleportacija na mjesto oznake. To ćemo ponašanje programirati u skripti Teleporter.cs koju ćemo zakvačiti za lijevi kontroler objekta CameraRig. Skripta mora sadržavati dvije javne globalne varijable: referencu na oznaku koju ćemo nazvati Pointer i akciju koja će pokretati teleportaciju (Slika 4.2.).

Te su akcije oblika SteamVR_Action_Boolean dohvaćene iz biblioteke SteamVR i pružaju nam jednostavnu mogućnost saznanja jeli na nekom od kontrolera neka akcija izvedena kao npr. klik na određeni gumb. U ovom je slučaju već predefinirano da je akcija naziva Teleport povezana s pritiskom kontrolera prema gore.



Slika 4.2. – Vanjski dijelovi skripte Teleporter.cs

Model oznake pod nazivom teleport_marker_mesh koju ćemo korisiti kako bi korisniku dali do znanja gdje će se teleportirati preuzeti ćemo također iz SteamVR paketa(Slika 4.3.). Taj ćemo model zapakirati u objekt u poslati ga kao referencu skripti.



Slika 4.3. – Model oznake teleportacije

Skripta Teleport.cs stvaramo na način da se u predefiniranoj metodi Update(), koja se poziva pri iscrtavanju svake nove sličice, provjeravamo jeli se pritisnula tipka prema gore, te u pozitivnom slučaju se poziva metoda TryUpdate(). Metoda TryUpdate(Slika 4.4.) najprije provjerava dvije stvari: jeli se trenutno teleportiramo

(varijabla isTeleportating) i je li teleportacija moguća (varijable hasPosition). Varijabla hasPosition se evaluira preko metode UpdatePointer() koja provjerava leži li oznaka na tlu na način da baca zraku u smjeru lijevog kontrolera i provjerava njezin sudar. Nakon toga dohvaćamo transformaciju(lokacija, rotacija i veličina) okvira kamere korisnika i lokaciju glave korisnika preko ugrađenih metoda SteamVR biblioteke. Poziciju korisnika na tlu groundPosition dobivamo tako da poziciji glave samo zamijenimo y komponentu za onu od okvira kamere. Onda stvaramo vektor translacije tako da oduzmemo lokaciju naše oznake lokacije i poziciju korisnika na tlu. Sada pozivamo rutinu MoveRig s predanom transformacijom okvira kamere i vektora translacije.

```
private void TryTeleport(){
    if(!hasPosition || isTeleporting){
        return;
    }

    Transform cameraRig=SteamVR_Render.Top().origin;
    Vector3 headPosition=SteamVR_Render.Top().head.position;

    Vector3 groundPosition=new Vector3(headPosition.x,cameraRig.position.y,headPosition.z);
    Vector3 translateVector=pointer.transform.position-groundPosition;

    StartCoroutine(MoveRig(cameraRig,translateVector));
}
```

Slika 4.4. – Metoda TryTeleport()

Rutina MoveRig (Slika 4.5.) zacrnjuje ekran na zadano vrijeme te se u međuvremenu primjenjuje translacija na lokaciju okvira kamere, te preko isTeleporting varijable onemogućava novu teleportaciju za to vrijeme.

```
private IEnumerator MoveRig(Transform cameraRig,Vector3 translation){
    isTeleporting=true;

    SteamVR_Fade.Start(Color.black,fadeTime,true);

    yield return new WaitForSeconds(fadeTime);
    cameraRig.position+=translation;

    SteamVR_Fade.Start(Color.clear,fadeTime,true);

    isTeleporting=false;

}
```

Slika 4.5. – Rutina MoveRig

4.2 Pokazivač u virtualnom prostoru

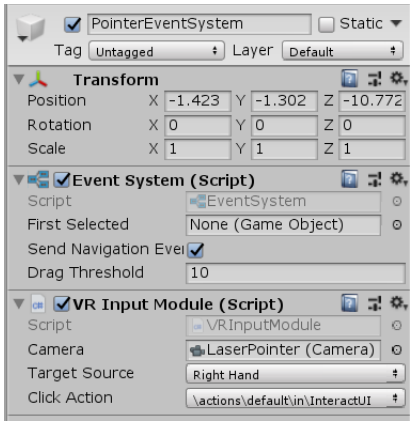
Iduća bitna stavka koja će služiti korisniku za interakciju u virtualnome prostoru je pokazivač. U tu svrhu stvorit ćemo objekt LaserPointer koji će tvoriti zraka i točka na vrhu zrake. Trenutno je potrebno samo staviti sfere u taj objekt i dodati mu opciju Line Render sa željenim parametrima, a sve ostalo potrebno oblikovat ćemo preko skripte Pointer.cs koju je potrebno dodati našem objektu. Kako želimo imati teleporter u lijevoj ruci, a pokazivač u desnoj, stvoreni objekt dodat ćemo kao dijete desnog kontrolera kako bi bio vezan za njegov lokalni koordinatni sustav.

Zadaća skripte Pointer biti će pravilo pozicioniranje sfere i pravilo iscrtavanje zrake(linije) koja pokazuje u smjeru kontrolera. Pri tome razlikovat ćemo dva slučaja. Prvi slučaj je kada pokazujemo u smjeru gdje nema niti jednog objekta koji će stati na put našoj zraki. Tada postavljamo duljinu te zrake na vrijednost unaprijed definiranu, a krajnju točku zbrajanjem početne pozicije i duljine. Drugi slučaj je onaj u kojemu zraka probada neki objekt u sceni. Tada ćemo dohvatiti lokaciju sudara s tim objektom i postaviti ju kao krajnju poziciju. Na kraju ćemo postaviti početak i kraj naše zrake temeljem dobivenih vrijednosti. Tako opisanu funkcionalnost želimo provjeravati kada god možemo pa ju stavljamo u metodu Update naše skripte (Slika 4.6.).

```
void Update(){

    PointerEventData data=inputModule.GetData();
    float targetLength=data.pointerCurrentRaycast.distance==0 ?
        defaultLength : data.pointerCurrentRaycast.distance;
    RaycastHit hit= CreateRaycast(targetLength);
    Vector3 endPosition=transform.position+(transform.forward*targetLength);
    if(hit.collider!=null){
        endPosition=hit.point;
    }

    dot.transform.position=endPosition;
    lineRenderer.SetPosition(0,transform.position);
    lineRenderer.SetPosition(1,endPosition);
}
```

Slika 4.6. – Update metoda skripte Pointer.cs

Međutim ovo do sada nam je bila samo vizualna reprezentacija, odnosno što se tiče interakcije s korisničkim sučeljem od ovoga nemamo nikakvu korist već samo da damo do znanja korisniku gdje je usmjerio pokazivač. Kako bi naše komponente korisničkog sučelja mogle reagirati na zraku moramo promijeniti sustav događaja tako da umjesto ulaza preko miša i tipkovnice koristimo ulaz naših kontrolera. Sustav događaja (eng. Event System) određuje na koji će način objekti u sceni primiti događaje temeljene na ulazima bili oni od miša, tipkovnice ili kontrolera. Za detekciju usmjerenja našeg kontrolera prema elementima korisničkog sučelja koristit ćemo grafički raspršivač zraka (eng. Raycaster). On nam služi kako bi sustav događaja znao gdje treba preusmjeriti trenutne ulazne događaje. Njegove je zadaća prikupiti sve potencijalne mete, shvatiti jesu li one pod određenim položajem i vratiti objekt koji je najbliži ekranu. Grafički raspršivač je korišten za elemente korisničkog sučelja, a postoje još i fizički 2D i 3D raspršivači korišteni za fizičke elemente. Trenutni sustav događaja mijenjamo tako da objektu u sceni koji nosi tu zadaću umjesto pretpostavljene skripte damo vlastitu (Slika 4.7.).



Slika 4.7. – Objekt koji predstavlja sustav događaja

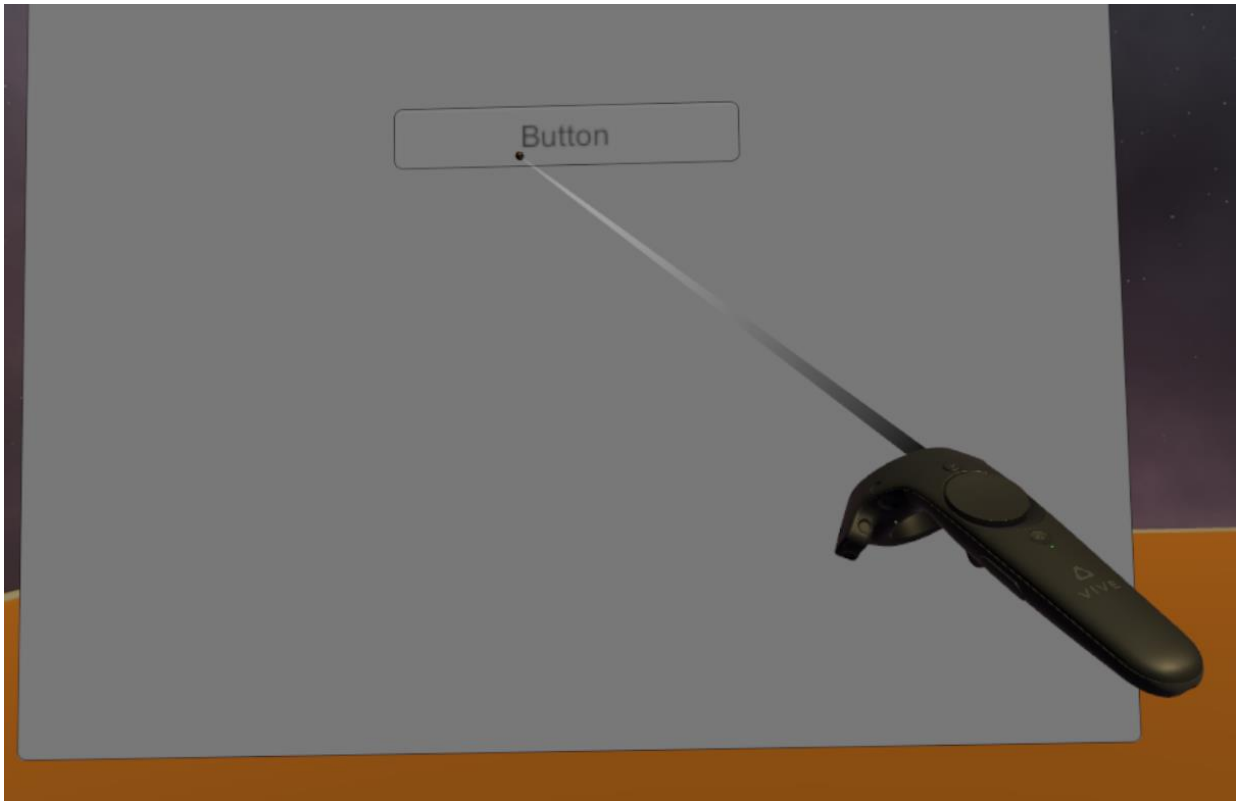
Sve potrebne funkcionalnosti smjestit ćemo u skriptu VRInputModule.cs. Kako ona opisuje sustav događaja naslijedit će sučelje BaseInputModule koje već sadrži neke metode za njegov opis. Opisat ćemo metodu Process() koja nosi glavnu odgovornost ove skripte (Slika 4.8.).

```
public override void Process(){

    try{
        data.Reset();
    }catch{}

    data.position=new Vector3(camera.pixelWidth/2, camera.pixelHeight/2);
    eventSystem.RaycastAll(data,m_RaycastResultCache);
    data.pointerCurrentRaycast=FindFirstRaycast(m_RaycastResultCache);
    currentObject=data.pointerCurrentRaycast.gameObject;
    m_RaycastResultCache.Clear();
    HandlePointerExitAndEnter(data,currentObject);

    if(clickAction.GetStateDown(targetSource)){
        ProcessPress(data);
    }
    if(clickAction.GetStateUp(targetSource)){
        ProcessRelease(data);
    }
}
```

Slika 4.8. – Metoda Process

U svakom ćemo ciklusu najprije resetirati podatke događaja pokazivača (eng. PointerEventData) u slučaju da smo nešto mijenjali u prethodnome. Zatim ćemo postaviti poziciju iz koje će naš raspršivač zraka dolaziti što će biti središte kamere objekta LaserPointer. Nakon toga odašiljemo zrake koristeći sve postavljene raspršivače zraka te uzimamo prvi objekt u koji je naša zraka udarila. Tom objektu poslat ćemo znak da naša zraka usmjerena u njega predefiniranom metodom HandlePointerExitAndEnter. Sada provjeravamo je li prijelaz za akciju koju smo odredili preko uređivača (akcija kojom će korisnik pritisnuti gumb) pozitivna (GetStateDown) ili negativna (GetStateUp) gdje je naš targetSource postavljen u uređivaču na referencu desnog kontrolera, te se u skladu s time pozivaju odgovarajuće funkcije. Te funkcije zajedničkom suradnjom provjeravaju implementira li trenutni objekt IPonterHandler sučelje, te na temelju toga pokreće akcije sučelja. O tim akcijama i samom sučelju ćemo nešto više reći u poglavlju 4.3 kada budemo izrađivali komponente grafičkog sučelja kojeg implementiraju.

Kako pokazivač u sceni izgleda prilikom prelaska preko gumba možemo vidjeti na slici 4.9.



Slika 4.9. – Pokazivač

4.3 Korisničko sučelje

Kako bismo korisniku omogućili upravljanje prikazima podataka potrebno je stvoriti izbornike. Sve naše izbornike ćemo stvarati na komponenti platna (eng. Canvas), te je njoj potrebno za svojstvo Canvas->Event Camera predati referencu na naš objekt LaserPointer opisan u poglavlju 4.2 kako bi bili u mogućnosti ostvariti interakciju korisnika s izbornikom.

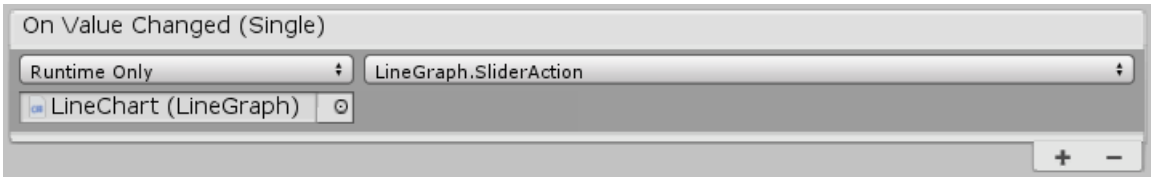
Prvo ćemo stvoriti izbornik s popisom prikaza podataka čijim se klikom stvara pojedini grafikon u sceni, te se korisnika stavlja ispred stvorenog grafikona. Za izgradnju izbornika Unity nam daje na izbor čitavu listu predefiniranih komponenata korisničkog sučelja koje samo treba posložiti u prostoru i stvoriti za svaki interaktivni element skriptu koja će opisivati njegovo ponašanje.

Na primjeru gumba za prikaz linijskog grafikona ukratko ćemo opisati njegovo ponašanje. Bitno je da skripta našeg gumba naslijedi sučelja IpointerEnterHandler, IpointerExitHandler i IpointerClickHandler iz kojih redom implementiramo funkcije OnPointerEnter, OnPointerExit i OnPointerClick. Ove funkcije nam omogućavaju modeliranje ponašanja kada pokazivač uđe u polje gumba i izađe iz njega, te kada kliknemo na gumb. Skripta sadrži i dvije varijable Color32 koje određuju boju kada pokazivač pređe preko gumba i standardnu. Na slici 4.10. dan je isječak funkcije OnPointerClick koja se poziva kada kliknemo na gumb. Rezultat ove funkcije je promjena boje gumba i prikazivanje odnosno skrivanje našeg linijskog grafikona temeljeno na prošlom stanju koje se pamti u globalnoj varijabli. Prikazivanje i skrivanje se temelji na aktiviranju odnosno deaktiviranju određenih komponenti grafikona koji određuju njegova fizička svojstva (prikazivanje i detekcija sudara). Također postavljamo zastavicu skripte grafikona active na true kako bi se osvježile njegove vrijednosti, kako bi evidentirali moguće korisnikove izmjene nad svojstvima grafikona u međuvremenu. Prilikom skrivanja grafikona još moramo paziti da uništimo sve njegove podkomponentne kako ne bismo višestruku iscrtavali grafikon.

```
public void OnPointerClick(PointerEventData eventdata){
    image.color=hoverColor;
    GameObject map=GameObject.Find("LineChart");
    if(lineChartFlag==false){
        map.GetComponent<LineGraph>().active=true;
        map.GetComponent<LineGraph>().enabled=true;
        map.GetComponent<BoxCollider>().enabled=true;
        lineChartFlag=true;
    }else{
        foreach (Transform child in map.transform) {
            GameObject.Destroy(child.gameObject);
        }
        map.GetComponent<LineGraph>().active=true;
        map.GetComponent<BoxCollider>().enabled=false;
        map.GetComponent<LineGraph>().enabled=false;

        lineChartFlag=false;
    }
```

Slika 4.10. – OnPointerClick metoda skripte LineChartButton.cs

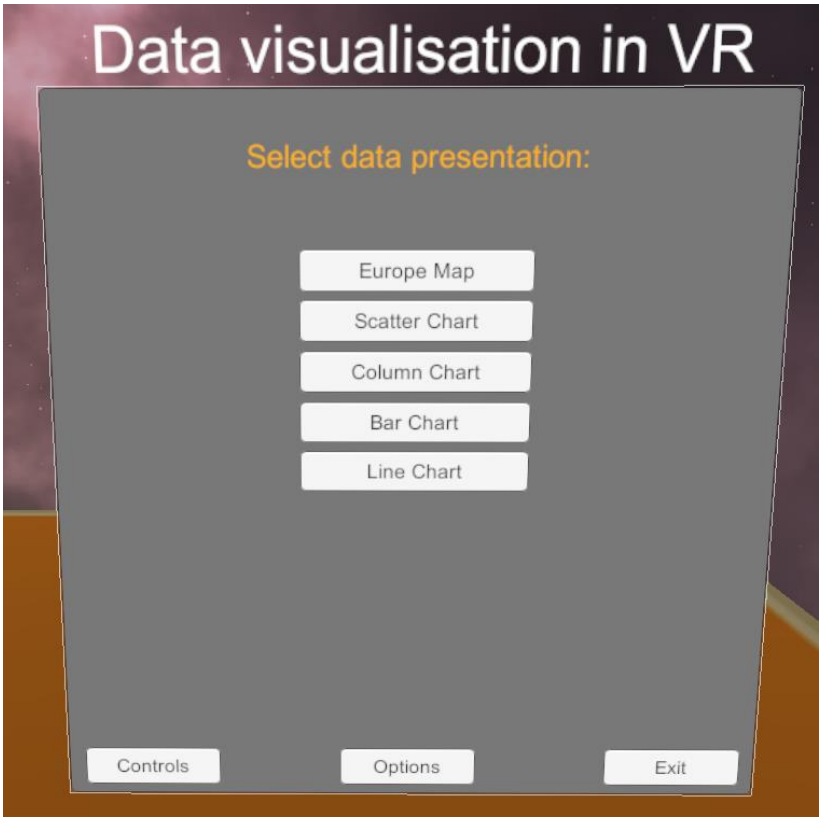
Drugi izbornik koji želimo napraviti je onaj u kojemu ćemo dati korisniku mogućnost izmjena svojstava (parametara) pojedine reprezentacije podataka. Parametre koje želimo dati korisniku na promjenu su: veličina koju mijenjamo kliznikom i datoteka iz koje čitamo podatke i stupce iz datoteke iz kojih dohvaćamo podatke preko padajućeg izbornika. Radi jednostavnosti implementacije ograničit ćemo se na .csv datoteke u direktoriju C:\VRdata. Kliznike se programirali tako da svaka skripta grafikona sadrži referencu na svoj kliznik, a svaki u svakom od kliznika preko uređivača postavljamo funkciju koja će se obaviti pri promjeni vrijednosti kliznika (Slika 4.11.).



Slika 4.11. – Postavke kliznika

Implementacija padajućeg izbornika se temelji na metodi Update() unutar skripte izbornika. U njoj se provjerava jeli se vrijednost izbornika u međuvremenu promijenila i na temelju toga mijenja parametre odgovarajućeg grafikona i poziva metodu osvježavanja istog.

Prikaz dva opisana izbornika u prostoru možemo vidjeti na slikama 4.12. i 4.13.



Slika 4.12. – Glavni izbornik

Slika 4.13. – Izbornik postavki

5. Interakcija u virtualnoj stvarnosti

Interakcije u virtualnoj stvarnosti smo se već kratko dotakli u poglavlju 4.2 kada smo spominjali kako korisnik može upravljati elementima korisničkog sučelja. U ovom ćemo poglavlju povezati poglavlja 3. i 4. na način da ćemo omogućiti korisniku interakciju s modelima prikaza podataka koje smo napravili. Točnije, moramo omogućiti korisniku da prelaskom točke ili stupca grafikona dobije uvid o točnoj vrijednosti tog elementa i omogućiti mu da pokretom kontrolera translatira i rotira grafikon u sceni.

5.1 Prikaz podataka

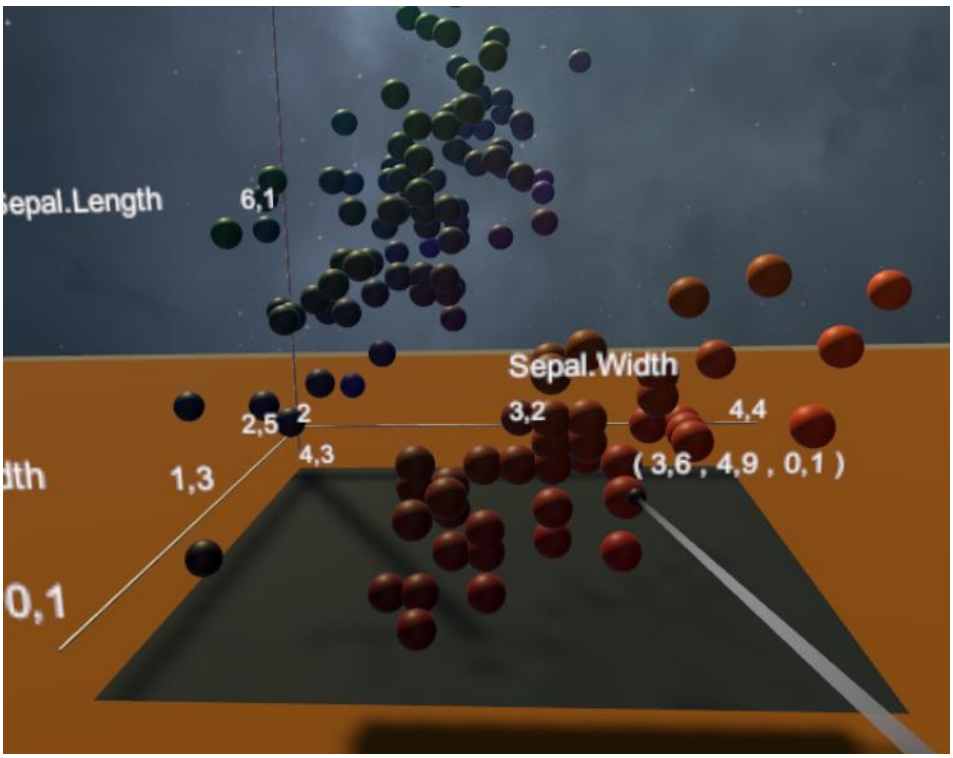
Korisniku ćemo dati uvid u konkretne podatke prikazujući brojčanu vrijednost kao tekst iznad odabranog elementa. Kada korisnik želi saznati točnu vrijednost neke točke u raspršenog grafikonu ili vrijednost stupca na karti Europe sve što treba je usmjeriti pokazivač prema tome elementu te će mu se ona prikazati iznad njega. Izrada ove funkcionalnosti nije pretjerano zahtjevna štoviše svodi se na proširivanje već napravljenih skripti. Najprije ćemo u skriptu svakog grafikona dodati metodu showData (Slika 5.1.) koja prima objekt u sceni kao argument. Ono što ova metoda radi je uzima ime poslanog objekta koje će odgovarati njegovoj vrijednosti te iscrtava tekst s formatiranom vrijednosti imena tog objekta i stavlja ga iznad objekta. Na kraju se sprema posljednji prikazani podataka u globalnu varijablu kako ne bismo više puta iscrtavali isti prikaz vrijednosti. U slučaju da se ovoj metodi pošalje neki neodgovarajući objekt, na primjer pozadina grafikona, njegovo se ime neće moći parsirati kao brojčana vrijednost, stoga će se izaći iz metode.

```
public void showData(GameObject child){
    if(lastShownData){
        Destroy(lastShownData);
        lastShownData=null;
    }
    var parts=child.name.Split(' ');
    string number=parts[parts.Length-1];
    try{
        float Number = float.Parse(number);
    }catch{
        return;
    }
    Vector3 pos=child.transform.position;
    GameObject dataName=Instantiate(graphText);
    Transform dataNameT=dataName.transform;
    dataNameT.parent=child.transform;
    dataNameT.localRotation=Quaternion.Euler(0,0,0);
    dataNameT.localPosition=new Vector3(0.2f*scale,0.5f*scale,0);
    dataName.GetComponent<TextMesh>().characterSize=0.025f*scale;


    dataName.GetComponent<TextMesh>().text = "( "+parts[0]+" , "+parts[1]+" , "+parts[2]+" )";

    lastShownData=dataName;

}
```

Slika 5.1. – Metoda showData

Slanje ispravnog objekta grafikonu oblikovat ćemo u skripti Pointer gdje se iscrtava zraka koju pokazivač odašilje u smjeru kojeg korisnik pokazuje. U toj skripti potrebno je proširiti metodu Update na način da ako je registrirano da je zraka udarila u neki objekt onda se ispita roditelj od objekta, ako je on neki od naših grafikona tada pokrećemo metodu showData na njemu s argumentom objekta u kojeg je udarila naša zraka. Ovime se ograničavamo da ćemo metodi showData slati samo elemente odgovarajućeg grafikona.

Prikaz vrijednosti točke raspršenog grafikona u koju je korisnik usmjerio pokazivač onako kako to korisnik vidi u aplikaciji prikazano je na slici 5.2.



Slika 5.2. – Prikaz vrijednosti preko pokazivača

5.2 Translacija i rotacija

Idući zahtjev koji moramo ispuniti je dati korisniku mogućnost pomicanja i okretanja grafikona bilo gdje u sceni. Time prepuštamo korisniku da samostalnu uredi svoje prikaze podataka kako mu najviše odgovaraju. Ovu funkcionalnost ćemo ostvariti tako da korisnik može prilaskom desnog kontrolera ishodištu grafikona držati pritisnutu stražnju tipku te će onda grafikon pratiti pomak kontrolera kao da je vezan za njega. Isto tako ako korisnik želi rotirati grafikon oko osi svoga kontrolera prići će ishodištu grafikona i držati bočne tipke te rotacijom ruke rotirati grafikon.

Odmah se može primijetiti kako trebamo na neki način proširiti objekt koji opisuje desni kontroler. Moramo mu dodati komponente sfernog sudarača (eng. Sphere Collider) i učvršćenog zgloba (eng. Fixed Joint). Sudarač nam služi kako bismo mogli registrirati da se kontroler nalazi u blizini ishodišta grafikona, jer je upravo njegova zadaća registrirati sve objekte koje dođu u njegovo osjetilno polje. Uloga

učvršćenog zgloba je vezati pomicanje jednog objekta za pomicanje drugog što je upravo ono što nama treba kada će korisnik držati pritisnutu tipku. Još ćemo desnom kontroleru dodati dvije skripte kako bismo oblikovali ponašanje translacije i rotacije, a to su HandPickUp i HandPickUpRotate.

Kako ne bismo morali ispitivati jeli objekt kojeg osjeti naš sudarač desnog kontrolera vrsta nekog prikaza podataka stvoriti ćemo apstraktno sučelje Interactable koje sadrži metode Update i Start koje će očito u Unity-u uvijek biti implementirane. Također sučelje zahtjeva da se primjeni na objektima s komponentom krutog tijela (eng. RigidBody). Takva komponenta je zahtijevana ako želimo objekta zakvačiti za neki učvršćeni zglob, a ona stavlja kretanje objekta pod kontrolu fizičkog pokretača Unity-ja. Na ovaj način ćemo omogućiti skriptama da pamte samo objekte koji su naslijedili sučelje Interactable (naši prikazi podataka).

U svakoj skripti čiji objekt posjeduje komponentu sudarača moguće je izvesti metode OnTriggerEnter i OnTriggerExit koje se pokreću kada sudarač osjeti neki objekt u svojoj okolini. Tim metodama se služimo kako bismo punili i praznili našu listu objekata koje nasljeđuju sučelje Interactable.

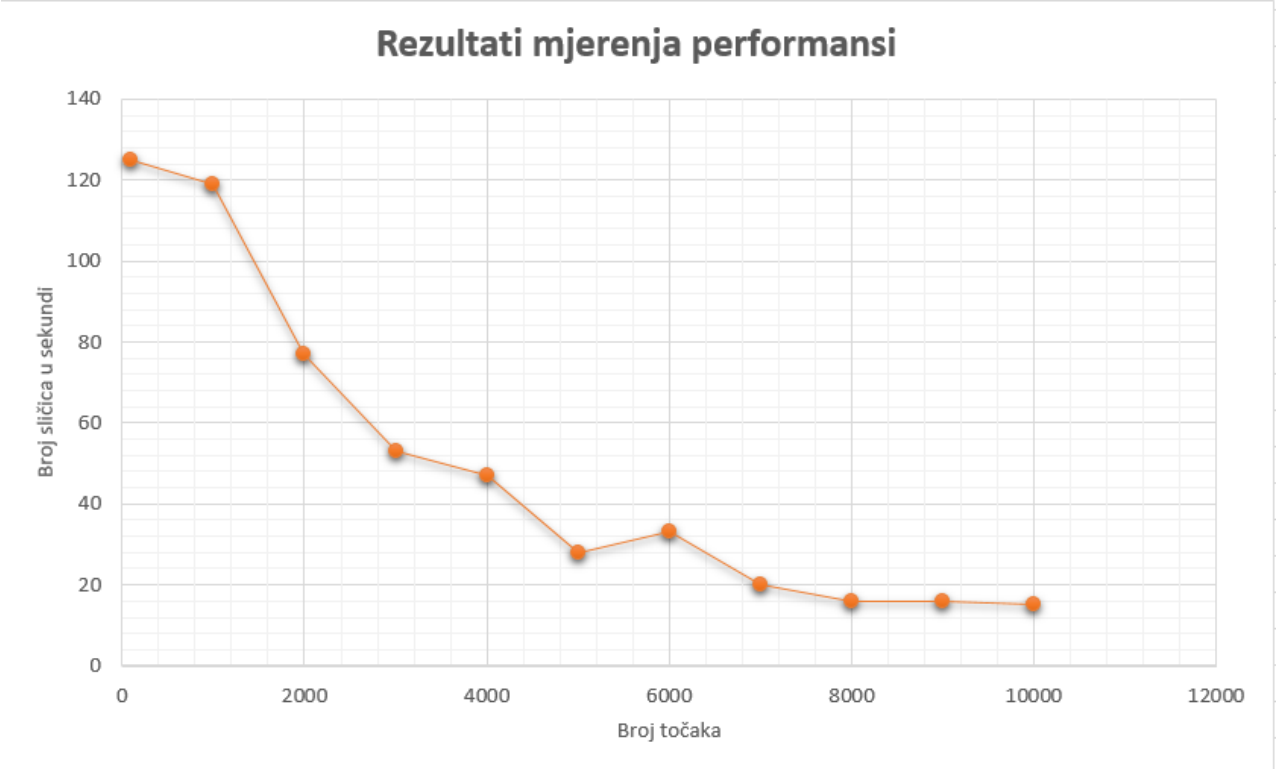
Skripta HadnPickUp neprestano očitava jeli se dogodila zadana akcija (u našem slučaju stražnja tipka) i koji se njezin prijelaz dogodio te na temelju toga zove funkcije Pickup odnosno Drop. Funkcija PickUp pretražuje listu svih objekata koje nasljeđuju sučelje Interactable, a osjetio ih je sferni sudarač, te nalazi najbliži objekt. Njegovu komponentu krutog tijela veže za učvršćeni zglob desnog kontrolera tako da sad prati korisnikove pokrete. U ovoj skripti još zamrzavamo rotaciju po svim trima osima krutog tijela grafikona kako bi osigurali da se provodi samo translacija. Kada korisnik otpusti zadanu tipku poziva se funkcije Drop koja otpušta vezani objekt od zgloba.

Skripta HandPickUpRotate je gotovo identična, a jedina je razlika što se nakon vezanja krutog tijela zamrzavaju pozicije grafikona, tako da osiguramo jedino provođenje rotacije.

6. Rezultati izvođenja

U sklopu rezultata izvođenja fokusirat ćemo se na raspršeni grafikon. Taj nam je tip grafikona najpogodniji za ispitivanje kako veličina skupa podataka naše .csv datoteke utječe na performanse programa. Upravo je ovom grafikonu najlakše predati veliki skup podataka jer se zasniva na prikazivanju sfernih objekata u sve tri dimenzije. Upravo smo iz ovog razloga tom grafikonu dodali i uvid u broj prikazanih točaka.

Korelaciju performansi i veličine skupa podataka najjednostavnije ćemo prikazati u odnosu broja točaka na zaslonu i broju sličica u sekundi koje je grafički procesor u stanju generirati u zadanom trenutku. Podaci u tu svrhu su prikupljeni preko Unity prozora sa statistikama. Sva mjerenja su izvršena na grafičkom procesoru Nvidia GTX 1070, a rezultati su dani grafikonom na slici 6.1.



Slika 6.1. – Rezultati mjerenja performansi

Kao što je vidljivo iz danog grafikona broj sličica u sekundi opada s brojem točaka raspršenog grafikona. Najveći pad se osjeti oko prelaska broja od 1500 točaka te oko tog broja i granica koja je preporučljiva za iskustva u virtualnoj stvarnosti (90 sličica u sekundi). Uzrok ovakvom padu performansi je dodavanje kompleksnosti, na već ovako zahtjevan prikaz u virtualnoj stvarnosti, za otprilike milijun vrhova na svakih novih tisuću točaka. Iz grafikona isto možemo zaključiti da je pad performansi logaritamski u odnosu na broj točaka. Valja napomenuti da veliki skupovi podataka osim pada broja sličica u sekundi uzrokuju i dodatno čekanje nakon slanja zahtjeva za njihov prikaz što je uzrokovano dugom obradom u MyCSVReader klasi i traženju minimuma i maksimuma vrijednosti za svaki stupac.

7. Zaključak

Po završetku valja se osvrnuti na motivaciju iz uvoda, vrednovati rezultate prikazanog rada i donijeti valjan zaključak. Odmah možemo uočiti kako smo ispunili glavnu točku uvodne motivacije, a to je prikaz trodimenzionalnih grafikona gdje je najbolji primjer za to raspršeni grafikon. Upravo je u virtualnoj stvarnosti takav prikaz najefikasniji jer ne samo da možemo rotirati i pomicati sam grafikon već se i mi možemo kretati oko njega i prilagoditi svoj kut pogleda. Od ostalih prikaza podataka uspješno smo modelirali linijski, stupčasti i trakasti grafikon te kartu Europe. Karte Europe iako nije pretjerano statistički i znanstveno intersantan prikaz, takva vrsta prikaza odlična je za korištenje putem raznih medija jer je njime najlakše prenijeti običnom čovjeku informaciju i opisati situaciju.

Druga stavka motivacije bila je interakcija s grafikonima. Interakciju smo ostvarili omogućivši korisniku slaganje grafikona u prostoru prema njegovom osobnom izboru korištenjem posebnih gumba za translaciju i rotaciju. Također smo mu omogućili da ima pristup vrijednosti svakog pojedinog podatka upiranjem pokazivača u element kojem želi saznati vrijednost.

Valja napomenuti kako je programsko rješenje djelomično univerzalno kroz druge platforme virtualne stvarnosti kao npr. Oculus VR ili neki od brojnih uređaja za prikaz virtualne stvarnosti na mobilnim uređajima koji su sve raširenij i cjenovno dostupniji korisnicima. Dio programskog rješenja koji nije prenosiv je onaj iz 4. i 5. poglavlja, odnosno koji se odnosi na posebnosti HTC Vive uređaja i SteamVR biblioteke kao što su korištenje posebnih kontrolera i slobodno kretanje u virtualnoj stvarnosti.

Kao nadogradnju u budućnosti istaknuo bih prikaz raznih dodatnih statističkih podataka koje se daju izračunati iz skupa podataka kao npr. standardna devijacija, kvartili i medijan, te moguća predviđanja kretanja nekih grafikona na temelju dostupnog skupa podataka. Bilo bi korisno i poboljšati učitavanje podataka iz datoteka na način da program sam prepoznaje tip podataka na temelju dane datoteke koji su prikazi podataka najbolji za dane podatke i optimirati proces učitavanja kako bismo mogli učitavati još veće skupove podataka.

8. Literatura

[1] Unity User Manual, 2019., Datum pristupa: 10.5.2019.
https://docs.unity3d.com/Manual/index.html

[2] SteamVR User Manual, Datum pristupa: 10.5.2019.
https://valvesoftware.github.io/steamvr_unity_plugin/api/Valve.VR.html

[3] HTC Vive povijest Wikipedija, Datum pristupa: 1.5.2019.
https://en.wikipedia.org/wiki/HTC_Vive

[4] HTC Vive značajke, Datum pristupa: 1.5.2019.
https://www.vive.com/us/product/vive-virtual-reality-system/

[5] Unity povijest Wikipedija, Datum pristupa: 1.5.2019.
https://en.wikipedia.org/wiki/Unity_(game_engine)

[6] Statistički podaci obnovljivih izvora energije Republike Hrvatske, 2016.
http://www.eihp.hr/wp-content/uploads/2018/06/EUH2016.pdf

[7] Statistički podaci Europske Unije, Datum pristupa: 20.4.2019.

https://ec.europa.eu/eurostat/statistics-explained/index.php/Main_Page

[9] Gustafsson M., Odd O. Virtual Reality Data Visualization Concepts, technologies and more, 2018.

https://www.diva-portal.org/smash/get/diva2:1222037/FULLTEXT02.pdf

[10] Beginning Dana Visualization in Unity: Scatterplot Creation , 2017., Datum pristupa: 10.5.2019.

http://sites.psu.edu/bdssblog/2017/04/06/basic-data-visualization-in-unity-scatterplot-creation/

Vizualizacija statističkih podataka upotrebom HTC Vive uređaja

Sažetak

Ovaj završni rad opisuje izradu različitih vrsta reprezentacije podataka u pokretaču Unity. Opisana je izrada četiri vrste grafikona: linijski, stupčasti, trakasti i raspršeni. Također je prikazano kako je moguće skupove podataka vizualizirati na geografskoj karti, u ovome slučaju Europe. Svi su ti postupci opisani u virtualnoj stvarnosti korištenjem HTC Vive uređaja i SteamVR biblioteke. Pokazani su koraci u izradi popratnog programa koji omogućava korisniku učitavanje proizvoljnog skupa podataka, odabir reprezentacije po želji, te kretanje i interakcija s objektima u stvorenoj virtualnoj sceni. Na kraju su razmotrena ograničenja i utjecaj veličine skupa podataka na izrađeno programsko rješenje.

Ključne riječi: vizualizacija podataka; virtualna stvarnost; HTC Vive; Unity; statistika; grafikoni

Statistical data visualization using the HTC Vive device

Abstract

This final work describes the creation of multiple types of data representation in Unity graphics engine. Four types of charts are described: line, column, bar and scattered graph. It is also shown how data sets can be visualized on a geographic map, in this case Europe. All these procedures are described in virtual reality using HTC Vive headset and SteamVR library. Taken steps required to create supporting software solution which allows user to load an arbitrary data set, select a representation as desired, move and interact with object in the created virtual scene are also shown. Finally, the limitations and impact of the data set size on the created software solution were discussed.

Keywords: data visualization; virtual reality; HTC Vive; Unity; SteamVR; statistics; graphs