SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2007

RAZVOJ IGARA U OKRUŽENJU DOPUNJENE STVARNOSTI

Adnan Abdagić

Zagreb, lipanj 2011.

Sadržaj

Uvod

Industrija video igara danas predstavlja najrašireniju i najprofitabilniju granu zabavne industrije. Razvojem novih tehnologija se konstantno traže novi načini kako bi se utjecalo na proširivanje čovjekovog doživljaja interakcije s računalnim svijetom.

Dopunjena stvarnost (engl. *Augmented Reality, AR*) je tema koja se trenutno veoma brzo rasprostire u raznim područjima, naročito u informiranju potrošača i interaktivnoj zabavi. Razlog tome su najviše nove generacije mobilne sklopovske opreme koje omogućuju njen prikaz.

U ovom radu bit će objašnjeni osnovni elementi neophodni u postupku izrade igara pomoću tehnologije dopunjene stvarnosti. Izradit će se i praktični prototip igre koja će demonstrirati dotičnu tehnologiju.

1. Osnove dopunjene stvarnosti

1.1. Teorijska podloga dopunjene stvarnosti

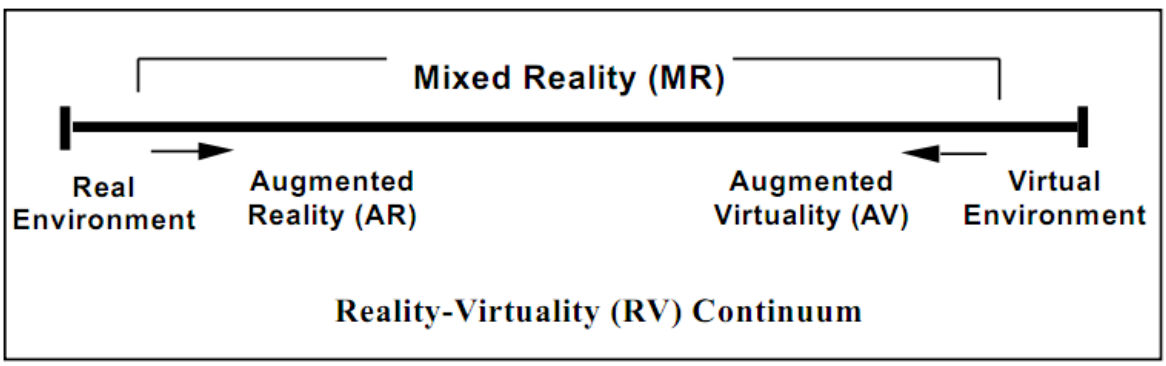
Pod dopunjenom stvarnošću podrazumijeva se proširenje stvarnog svijeta s virtualnim. To se dešava tako da se u stvarnom vremenu dodaju 2D ili 3D elementi virtualnog okruženja u video sliku, i time proširuje korisnikovo viđenje svijeta [11].

Osnovne karakteristike dopunjene stvarnosti su:

1. Kombinacija stvarnog i virtualnog
2. Interakcija u stvarnom vremenu
3. Poravnavanje u 3D

Međutim, moramo razlikovati pojedine razine stvarnosti, koje se najbolje opisuju konceptom virtualnog kontinuuma stvarnosti (*Reality-Virtuality continuum*) – Milgramov kontinuum [1]. On se predstavlja kontinuiranom skalom u rasponu od (potpuno) virtualnog do stvarnog okruženja (Slika 1.1). Prema Milgramu postoji nekoliko razina stvarnosti:

1. *Virtual Reality* (VR) – virtualna stvarnost
2. *Augmented Virtuality* (AV) – prošireni privid
3. *Mixed Reality* (MR) – mješovita stvarnost
4. *Augmented Reality* (AR) – proširena/dopunjena stvarnost
5. *Real Enivonment* (*Reality*) – stvarnost



Slika 1.1 Milgramov kontinuum stvarnog i virtualnog

1.1.1. Paradigme i problemi dopunjene stvarnosti

Dvije česte paradigme [2] za dopunjenu stvarnost su (Slika 1.2):

- *Magic Mirror* (magično ogledalo)
- *Magic Lens* (magični objektiv)

Magic mirror tehnika uključuje stavljanje nekog oblika ekrana (monitor, televizija, projektor i slično) iza područja koje se snima AR video kamerom. Zaslon je na taj način ogledalo, i često prikazuje proširenja stvarnosti preko videa u stvarnom vremenu.

Magic lens pogled je nešto drugačiji pristup. Umjesto da ponudi ogledalo, ovaj pristup omogućuje korisniku da vidi sliku stvarnog svijeta proširenu s AR elementima.

Slika 1.2 *Magic Mirror* (lijevo) i *Magic Lens* (desno)

Dopunjena stvarnost teži da korisniku pruži izuzetno jednostavan i intuitivan pristup podacima. Problemi koje treba riješiti za ispravnu realizaciju dopunjene stvarnosti su miješanje, poravnavanje slike, te prikupljanje podataka.

Miješanje slike predstavlja istovremeni prikaz stvarne i virtualne slike. Razlikujemo četiri tipa miješanja slike (vizualizacije):

1. *Optical See Through* (OST) – optičko miješanje
2. *Video See Through* (VST) – video miješanje
3. *Monitor Augmented Reality* (MAR) – proširena stvarnost na zaslonu
4. *Projective Augmented Reality* (PAR) – projekcijska proširena stvarnost

Kod optičkog miješanja korisnik gleda kroz polu-prozirno ogledalo, i na taj način vidi sliku stvarnog svijeta iza ogledala koja se proširi s dodatnim grafičkim elementima generiranim od strane računala u stvarnom vremenu. S druge strane, kod video miješanja korisnik gleda samo jednu video sliku stvarnog svijeta koju snima kamera – te se ona proširi dodatnim elementima. Oba ova načina miješanja zahtijevaju da korisnik na glavu nosi uređaj za prikaz (i snimanje), tako da je pogled okoline uvjetovan pokretima glave (Slika 1.3).

Slika 1.3 Optičko miješanje (lijevo) i video miješanje (desno)

Najčešći tip miješanja slike je proširena stvarnost na zaslonu, koja je zapravo video miješanje u kojem nije potrebno da korisnik uređaj za prikaz (*Head-Mounted Display*, HMD) nosi na glavi. Na taj način je slika uvjetovana pozicijom kamere, a prikaz se očitava na ekranu.

Projekcijska proširena stvarnost se postiže pomoću jednog ili više projektora, kojim se virtualna scena projicira u stvarni svijet [8].

Poravnavanje (engl. *registration*) osigurava da se virtualni predmeti točno poklapaju sa stvarnima. Za to je potrebno znati točan položaj i orijentaciju korisnika i svih bitnih predmeta u sceni. Poravnavanje u 3D znači da su virtualna proširenja dodaju u sliku s preciznom geometrijskom korelacijom sa scenom koja se vidi na zaslonu. Radi se o 3D iscrtavanju predmeta u koordinatnom sustavu koji je poravnat sa stvarnim svijetom, te se mogu postići i efekti poput realističnih prekrivanja predmeta (nije samo jednostavno miješanje slike). Za to se koriste različite tehnike praćenja (engl. *tracking*).

Napokon, korisne podatke koje želimo prikazati u proširenoj stvarnosti potrebno je prikupiti. Ovo prikupljanje podataka (engl. *sensing*) je vrlo širok skup tehnika, od ultrazvuka u medicini do prijenosa podataka s nacrta ili iz baza podataka.

1.1.2. Markeri i njihovo praćenje

Sustavi za dopunjenu stvarnost obično zahtijevaju neku indikaciju gdje točno se treba proširiti digitalna slika. To je najčešće postignuto s AR markerima, i metodom praćenja računalnim vidom i obradom snimaka. Pokušani su različiti tipovi markera, od light-emiting dioda (LED) do ljudskih ruku, međutim najjednostavniji oblik markera je jedinstveni uzorak vidljiv AR kameri kojeg može identificirati programska podrška sustava za dopunjenu stvarnost. Uzorci se fizički dodaju u stvarni svijet. Markeri se koriste i da bi se utvrdilo gledište kamere tako da bi se virtualni objekt prikladno renderirao.

Kada programska podrška za dopunjenu stvarnost prepozna marker, računalo može utvrditi poziciju i kut markera. Nakon te obrade, programska podrška postavi virtualni objekt preko sloja slike s kamere, te se proširena slika prikazuje korisniku. Ako AR programska podrška ispravno renderira objekt, prikazat će se kao dio scene stvarnog svijeta. Kako se i orijentacija računa, virtualni objekt se može postavljati kako se marker miče okolo u 3D prostoru.

Moguće je stvoriti efekt dopunjene stvarnosti i bez markera - to je poznato pod nazivom "*markerless AR*". Takvim sustavima nije potrebno dodavanje markera u scenu. Sustav za praćenje se može namjestiti da prati druge stvari, kao npr. LED-ice, reflektivne loptice i slično, međutim AR bez markera omogućuje da se kao markeri koriste prirodne značajke koje već postoje u slici, npr. kut prozora ili slika na zidu.

U budućnosti će se AR bez markera vjerojatno ispostaviti kao preferirana metoda. Međutim, dotična tehnika nije još dovoljno napredna da bi javnosti bilo moguće jednostavno koristiti tehnologiju, niti postoje sustavi bez markera koji daju bolje performanse od sustava s markerima.

1.1.3. Sklopovska oprema za dopunjenu stvarnost

Zadatak AR-sustava je realni svijet proširiti ili izmijeniti informacijama generiranim od strane računala. Da bi to postigli potrebni su nam određeni vanjski uređaji, koji razvojem tehnologije postaju sve kompaktniji i efektivniji u obradi informacija [5].

AR-sustav se sastoji od pet važnijih sklopovskih-sustava:

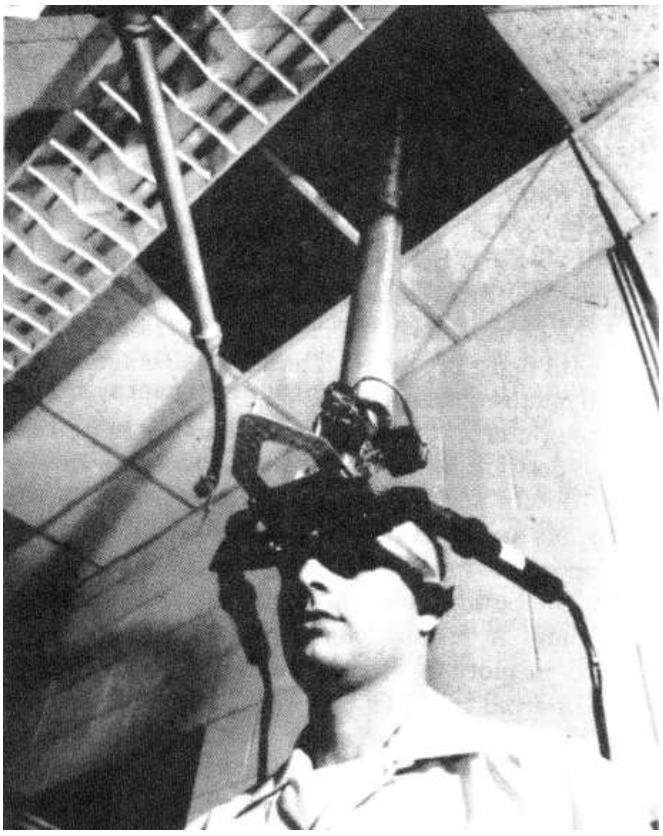
- Računalo za obradu informacija (ostvarenje prikaza i miješanje)
- Sustav za prikaz – ekran ili Head-Mounted-Display (Slika 1.4)
- Sustav za sinkronizaciju i praćenje
- Senzori za snimanje (kamera)
- Uređaji za unos (tastatura, miš)

Slika 1.4 Primjena suvremenih HMD-a

1.2. Podrijetlo dopunjene stvarnosti

U 1968. godini, Ivan Sutherland je napravio praktičan prototip nečeg što se široko smatra prvim sustavom za virtualnu stvarnost i prvim sustavom za dopunjenu stvarnost. Iako je koristio jednostavnu grafiku, taj projekt je bio geneza AR-a. Sutherlandov sustav je zahtijevao da korisnik nosi nezgrapan i glomazan HMD (Slika 1.5). Kako je bilo potrebno sa stropa mehanički spustiti na korisnika, prikladno je nazvan "Damaklov Mač" (engl."*Sword of Damocles*") [6].

Sutherland je prepoznao da mu je sučelje ograničeno pa je nastavio rad na boljim sustavima. Nekoliko godina nakon toga, Sutherland je u članku "*The Ultimate Display*" predstavio ideju i viziju savršenog sučelja - soba u kojoj isprogramirane stvari postaju stvarne [7].

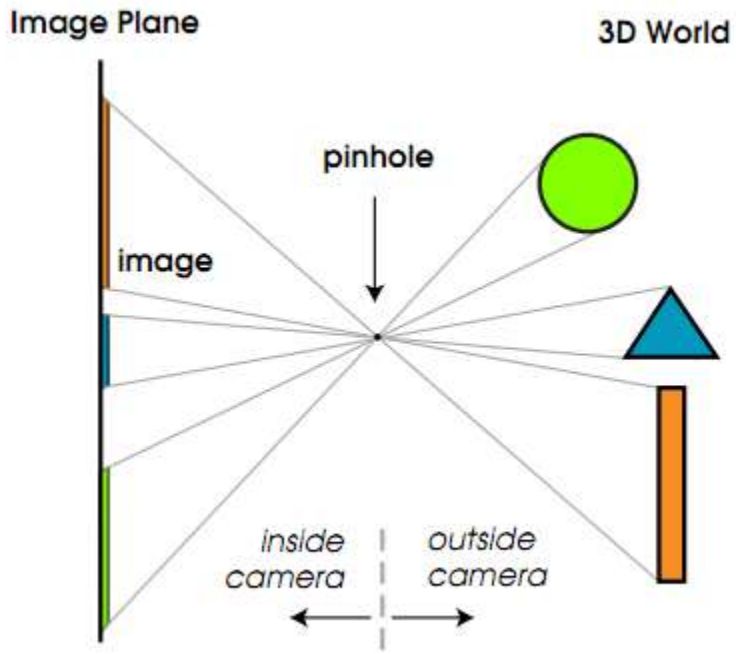


Slika 1.5 *Sword of Damocles*

1.3. Osnove računalne grafike za dopunjenu stvarnost

Programeri i umjetnici koji razvijaju AR aplikacije moraju uvijek biti svjesni dva osnovna principa:

1. Slike na televiziji, računalnim monitorima, PDA-ovima, i drugim ekranima su 2D reprezentacije 3D svijeta. Svi to znamo ali kako naš um napravi mentalnu scenu kad vidimo 2D sliku, važno se sjetiti da su slike i video samo 2D polja piksela.
2. Računalna grafika općenito koriti jednostavni *pinhole* model projekcije kamere. U tom modelu, svjetlost objekata u okolini prolazi kroz jednu točku u prostoru i pogađa ravninu slike koja je obično ravna (Slika 1.6).



Slika 1.6 Formacija slika s pinhole modelom

Prva točka je očita, ali je važno napraviti razliku između regularnih 2D slika i 3D slika - kao što su RADAR, LIDAR, SONAR, MRI ili CT slike. Takve 3D slike su primjerci 3D polja brojeva. Postoje i kamere koje hvataju samo 1D slike, i tehnički govoreći, jedan foto-senzor poput onog za mjerenje svjetlosti je 0D senzor, ali ćemo se baviti samo najčešćim slikama - 2D slikama.

Unutar računala ili digitalne kamere, slika je samo 2D polje brojeva. Cilj računalne grafike je napraviti takvo 2D polje da naš um vjeruje da gleda kroz prozor u 3D svijet. Moramo poštivati geometrijska pravila formiranja slike ukoliko želimo pravilno izgraditi scenu u našoj glavi. Naša podsvijest uvijek može razlikovati dobru i lošu geometrijsku formaciju. U dopunjenoj stvarnosti se to proširuje na vjerovanje da se računalno ostvaren objekt može pojaviti u slici stvarnog svijeta.

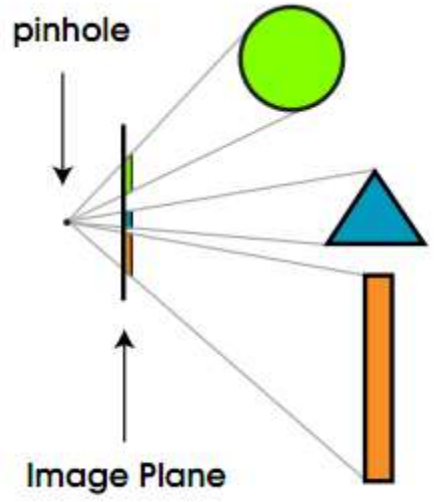
Druga fundamentalna točka - *pinhole* projekcija - nije toliko očita. Za razliku od realnih kamera, koje imaju leća, virtualne kamere korištene u računalnoj grafici su samo rupice (engl. *pinholes*) odnosno žarišne točke ili centri projekcije. To ne bi bilo praktično u stvarnom svijetu, ali u dopunjenoj stvarnosti se radi s takvim *pinhole* kamerama.

Izgrađujemo kameru s *pinhole* modelom kamere. Ovaj model definira osnove geometrijskog projiciranja slike kojim se karakteristike objekta preslikavaju na 2D ravninu. Predstavlja pojednostavljen model stvarne kamere iz optike i često se koristi u računalnoj grafici i računalnom vidu da opiše formacije slika u kameri.

Slika se obično hvata na ravnoj površini, koja je poznata pod nazivom ravnina slike. U filmskim kamerama ravnina slike je fotografski film, ali za elektroničko hvatanje slike, ravnina je polje regija osjetljivih na svijetlost u CCD ili CMOS kamerama. Svaki element na tim čipovima mjeri intenzitet svjetlosti za jedan piksel slike.

Slika koja se hvata *pinhole* modelom se naziva perspektivna slika. Tehnički gledano, takva slika je perspektivna projekcija, ili *pinhole* projekcija, jer se scena koja se hvata iscrtava iz perspektive rupice (engl. *pinhole*). Centralna os je pravac od žarišne točke okomita na ravninu slike, okrenuta je u smjeru gdje kamera gleda.

Kao kod stvarne kamere, u virtualnoj pinhole kameri svjetlost prvo prolazi kroz žarišnu točku i onda pogađa film ili senzor slike. To znači da je formirana slika izokrenuta naopako i unatrag. Ako ravninu slike postavimo ispred žarišne točke tako da je desna strana gore, onda će linija povučena od objekta kroz žarišnu točku presjeći tu desnu stranu ravnine u istom mjestu kao da je ravnina površine naopako i na istoj udaljenosti iza žarišne točke. Na istom principu funkcionira i ljudsko oko. Međutim, ista geometrija vrijedi i kad bi ravnina slike bila ispred rupice, i onda ne trebamo razmišljati o invertiranim slikama (Slika 1.7).



Slika 1.7 Ravnina slike je ispred rupice

Umjetnici ponekad koriste sličnu metodu da stvore perspektivnu sliku. Obično postave transparentno platno ili platno koje se jednostavno može maknuti van vida - te držeći glavu mirno, gledaju na jedno oko i slikaju na platno objekt koji im je u vidokrugu.

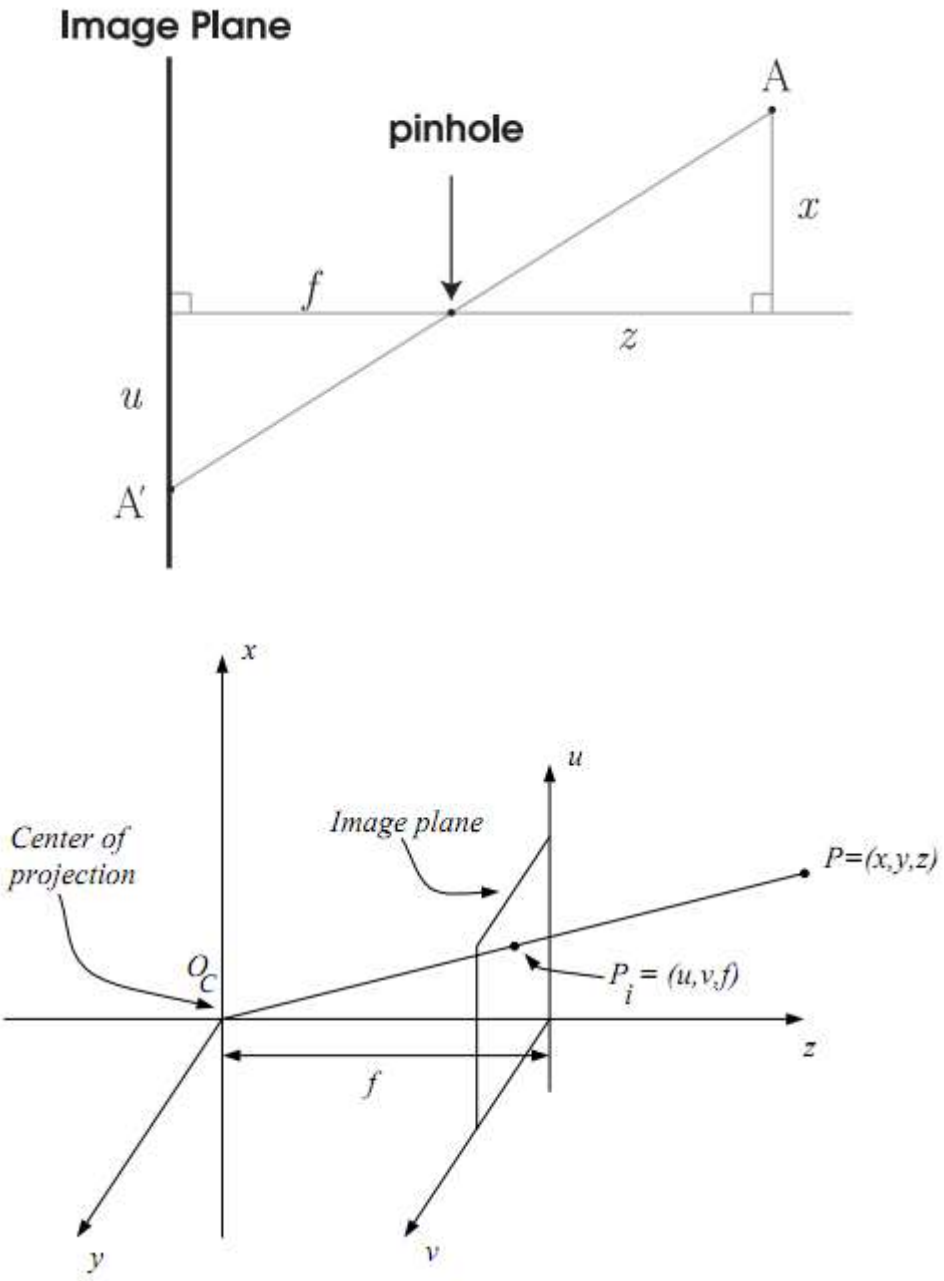
Leća, ili skupina leća se u stvarnim kamerama koriste kako bi se uhvatila veća količina svjetlosti. Leća uvode pojmove dubine vidnog polja i zamućenje, međutim te stvari se obično ne razmatraju u računalnoj grafici, naročito u onoj generiranoj u stvarnom vremenu, npr. u igrama i AR sustavima. Ponekad se takvi efekti simuliraju promjenom operatora zamućivanja na slike stvorene *pinhole* metodom. Objekti na različitim dubinama se obično prikazuju posebno i zamagljeno prije nego se povežu u jednu sliku (stvara efekt različitom količinom zamagljenja na različitim dubinama).

Za lakšu vizualizaciju, ravnina slike se može zamisliti ispred rupice tako da slika nije naopako. Ako imamo krug u 3D prostoru i hvatamo ga koristeći *pinhole* projekciju, možemo dobiti krug, ili možemo završiti s drugačijim ovalnim objektom. Ako krug nije paralelan s ravninom slike, onda se može prikazati elipsom. S druge strane, ako imamo ravnu crtu u 3D svijetu, uvijek će se projicirati u ravnu crtu u slici. Poligoni su ravne regije koje imaju segmente ravnih linija za rubove. *Pinhole* projekcija mora projicirati ravne linije u 3D svijetu u ravne linije u slici, tako da će se poligoni uvijek projicirati u poligone. Većina sustava za računalnu grafiku u stvarnom vremenu koriste poligone - koji se pružaju između tri ili četiri točke.

Postoje različiti načini uspostavljanja koordinatnih sustava, a mi ćemo koristiti Kartezijev gdje je centar projekcije na izvoru i ravnina slike je od njega na udaljenosti f (žarišna duljina).

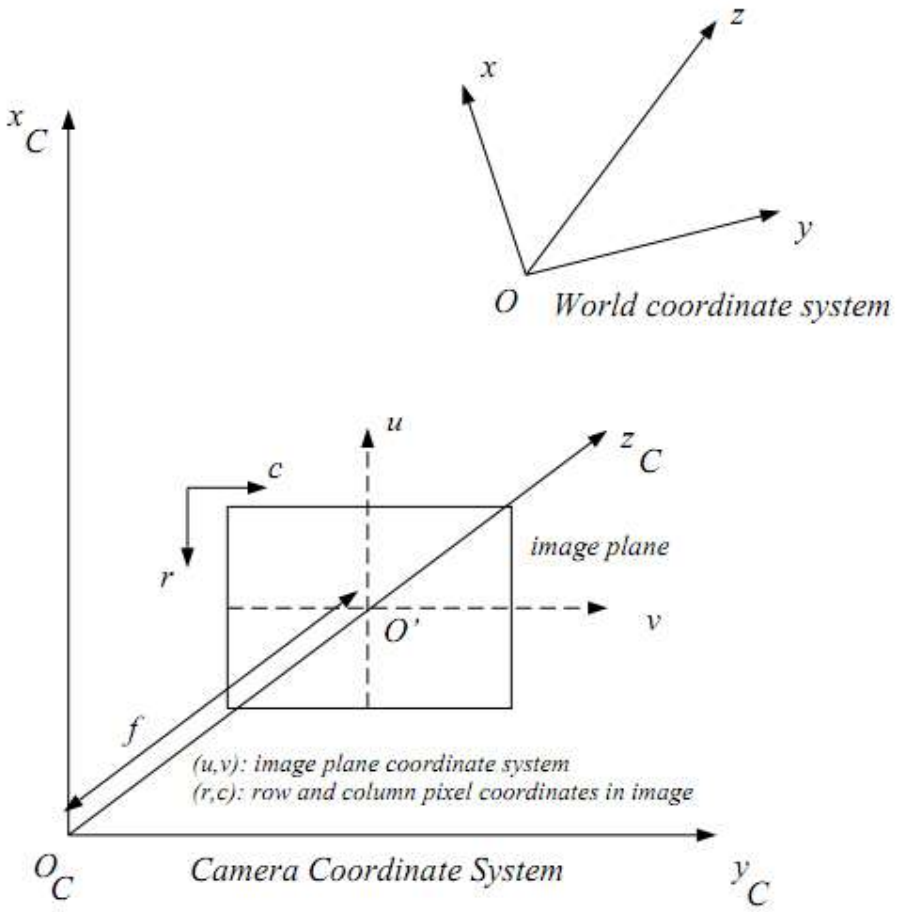
Slika točke u 3D-u se formira prolaskom zrake kroz točku i centar projekcije, O_c, te se onda izračuna presjek zrake i ravnine slike. Jednadžbe su zasnovane na principu sličnosti trokutova (Slika 1.8).

$$u = \frac{f * x}{z} \qquad\qquad v = \frac{f * y}{z}$$



Slika 1.8 *Pinhole* projekcija – matematički prikaz

Model kamere koji se koristi za generiranje grafičkih slika mora biti u vezi s koordinatnim sustavom piksela uređaja za prikaz. Odnos koordinata ekrana, *pinhole* modela i koordinata svijeta (Slika 1.9) mora se jasno definirati kako bi se uspostavio precizan odnos između lokacije piksela u slici i 3D zrake koja se projicira do te točke [14].



Slika 1.9 Međuodnosi različitih koordinatnih sustava

Pozicija i orijentacija kamere respektabilno prema koordinatnom sustavu svijeta (O, x, y, z) se definira krutom transformacijom koja se sastoji od rotacije R i translacije T.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \boldsymbol{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \vec{T}$$

Konačno, koordinate piksela su u vezi s koordinatama ravnine slike sa sljedećim jednadžbama (r označava redak, c stupac).

$$r - r_0 = s_u u$$

$$c - c_0 = s_v v$$

1.4. Igre i dopunjena stvarnost

Desetljećima razvoj video igara žudi za većim osjećajem uranjanja u virtualni svijet, te iako je napravljen velik napredak, tipična igra se još uvijek igra u statičnom položaju gledajući u ekran. Igre ne trebaju biti toliko pasivne i oslobađanje igrača je samo jedan od ciljeva AR tehnologije [12].

Ambiciozni cilj AR tehnologije je stvoriti osjećaj da virtualni objekti postoje u stvarnom svijetu. Kako bi se postigao taj efekt, programska podrška kombinira elemente virtualne stvarnosti (VR) s elementima stvarnog svijeta. Dopunjena stvarnost redovito uključuje proširenje digitalne video slike s 2D ili 3D objektima. Najjednostavniji primjer dopunjene stvarnosti je preslikavanje obične 2D slike na digitalni video. Međutim, moguće je dodati i 3D objekte - mogu se renderirati tako da se čini kao da pripadaju sceni koja sadrži 3D objekte.

Kad se virtualni objekti dodaju u scenu, to zovemo vizualna dopunjena stvarnost. Po definiciji, elementi dopunjene stvarnosti nisu vidljivi golom oku, tako da se vizualna dopunjena stvarnost oslanja na neki način prikaza. To može biti jednostavan monitor računala ili TV ekran, ili pak nešto više napredno – npr. HMD. Nove opcije prikaza se javljaju kako se znanstvenici više fokusiraju na razvoj sučelja na mobilnim uređajima, razvoj kamera i naprednijih HMD-a.

Koristeći AR moguće je dostići stupanj uranjanja koji je iznad onog čime ljudi asociraju igre [10] [9]. Međutim, to nije jedina uporaba AR tehnologije danas.

Slika 1.10 *Marble Game* od Ohan Oda

1.5. Primjeri primijenjene dopunjene stvarnosti

AR brzo dobiva na popularnosti, međutim to je još uvijek mlada tehnologija. Danas se najviše koristi u akademske svrhe, ali postoje i neke komercijalne primjene. Te primjene uključuju sve od arhitekture, edukacije, pa do medicine. Na osnovu različitih primjena jasno je da AR ima obećavajuću budućnost [3]. U sljedećem dijelu ćemo proći kroz neke istaknute primjere suvremenih AR projekata i mogućih područja njihove primjene [4].

1.5.1. Igre i zabava

Kao što je ranije rečeno, najveća primjena dopunjene stvarnosti je u industriji igara, međutim uveliko se koristi i u drugim dijelovima zabavne industrije - prije svega u televiziji. Na određene markere se ubacuju reklame ili se gledateljima daju dodatne informacije o programu. Recimo, sportski prijenosi su izvukli veliku korist od dopunjene stvarnosti (pak u hokeju kao marker, dresovi igrača, plakate oko igrališta i slično).

Slika 1.11 ARDominos (gore) i AR u televiziji (dole)

1.5.2. Obrazovanje

U kombinaciji s drugim tehnologijama, kao što je WiFi, dopunjena stvarnost se koristi kako bi prenijela dodatne informacije korisniku. Tako u edukacijske svrhe, sustavi za dopunjenu stvarnost su korišteni za izradu panoramske rekreacije historijskog događaja preko lokacije u stvarnom svijetu gdje se odigrala. Neki muzeji koriste tehnologiju kako bi prikazali detaljne modele nekih objekata (npr. srednjovjekovni dvorac - jednim klikom možemo skočiti kroz vrijeme i vidjeti kako on izgleda za 500 godina). Studenti se služe dopunjenom stvarnošću da dobiju veće razumijevanje stvari poput formacije oblaka, strukture svemira i galaksije, itd. kroz realistične simulacije.

Creative Research and Development team BBC-a je u 2006. godini pokrenuo projekt razvoja programske podrške koja je trebala komplimentirati nastavni plan i program Ujedinjenog Kraljevstva. Izradili su nekoliko interaktivnih priča za djecu pet do sedam godina koje su multimedijskim sadržajem dopunjavali knjige i slikovnice dok su ih čitali ispred računala (u knjigama i slikovnicama su bili određeni markeri).

Slika 1.12 Primjeri primjene AR u obrazovanju

1.5.3. Sigurnost i obrana

Vojska, konkretno Office of Naval Research and Defense Advanced Research Projects Agency (DARPA), je jedan od originalni pionira sustava dopunjene stvarnosti. Jedna od glavnih njeni primjena u vojsci je što nudi vojnicima na terenu presudne informacije o njihovom okruženju (pokrete prijateljskih i neprijateljskih jedinica). Dopunjena stvarnost će također ubuduće igrati veliku ulogu kod obavještajnih službi i agencija za provođenje zakona. Omogućit će policajcima da imaju potpuno detaljan pogled i informacije o stanju kriminala i patrolama u nekom području.

Avion F-35 Lightning II pilotu informacije prikazuje preko HMD-a uz pomoć tehnologija dopunjene stvarnosti. To je najrazvijeniji HMD današnjice.

Slika 1.13 Pilotski HMD u F-35 (lijevo) i mobilni HMD za pješadiju (desno

1.5.4. Medicina

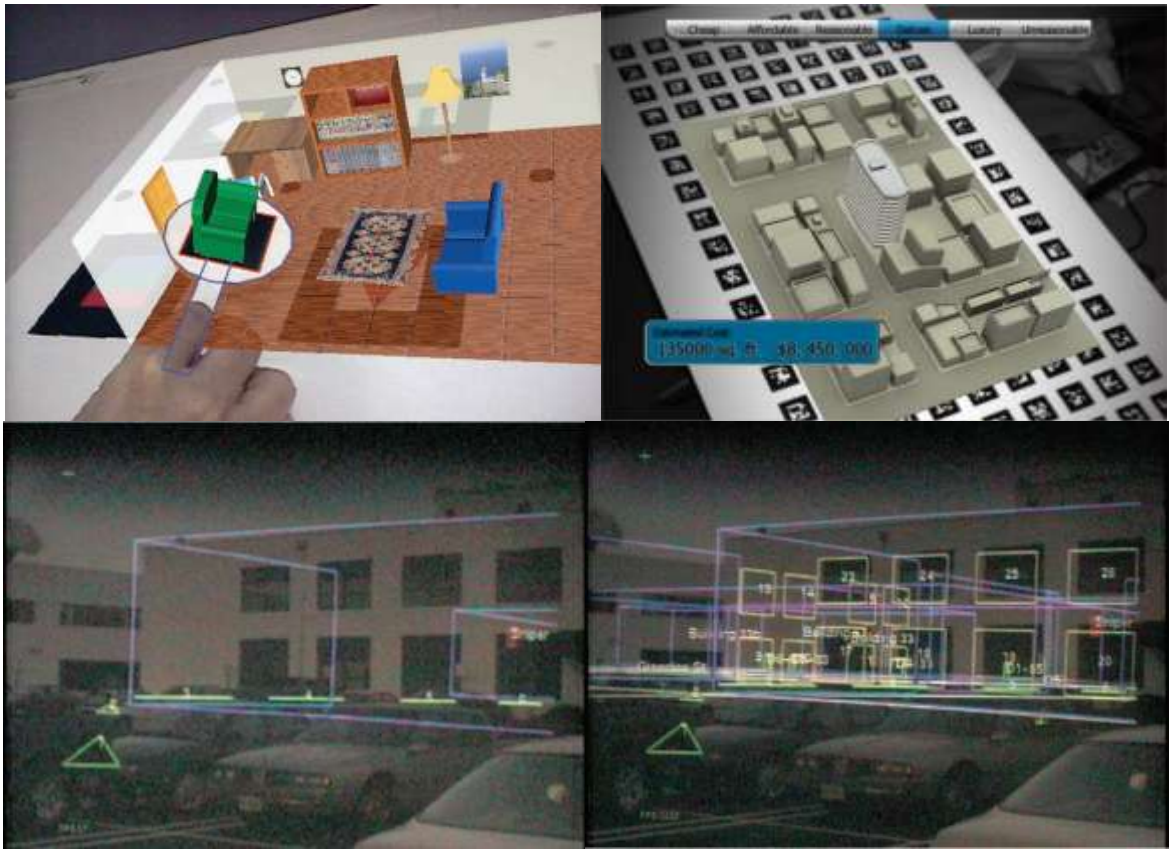
U medicini se dopunjena stvarnost može koristiti kako bi kirurgu pružila bolju čulnu percepciju tijela pacijenta prilikom operacije. To će rezultirati u operacijama s manje rizika i većom učinkovitošću. Sustav bi se također mogao koristiti zajedno s drugim medicinskim uređajima poput rendgena ili MRI-a kako bi doktoru odmah dao informacije koje treba za medicinsku dijagnozu ili odluku.

Još jedan karakterističan primjer je ultrazvuk pomoću dopunjene stvarnosti.

Slika 1.14 Medicinske primjene AR tehnologije

1.5.5. Arhitektura i dizajn

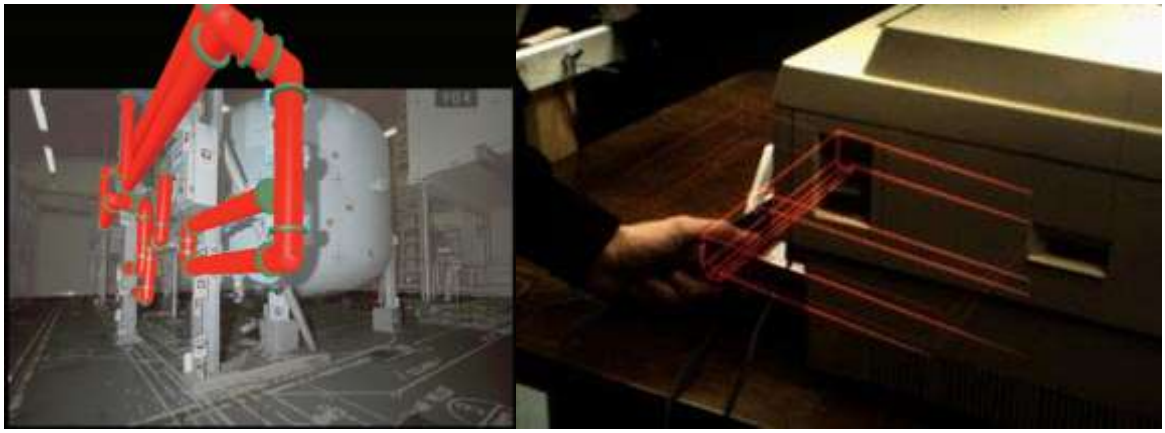
Pomoću dopunjene stvarnosti se mogu simulirati konstrukcijski planovi iz arhitekture, građevine i dizajna. Takve sustave koriste arhitekti, kao i njihovi klijenti za simuliranje uređenja nekog prostora, ali i za simuliranje izgleda cijelog konstrukcijskog plana. Veliku primjenu ima i u nadgledanju i popravljanju raznih instalacija, gdje pomoću AR-a možemo utvrditi gdje se nalaze sve instalacije kako bi izolirali onu koju moramo eventualno popraviti.



Slika 1.15 AR u arhitekturi. Dizajn (gore), prikaz instalacija (dole).

1.5.6. Industrijska proizvodnja, održavanje i upravljanje

Dopunjena stvarnost se može koristiti da se usporede digitalni i fizički modeli postrojenja kako bi se brzo našla odstupanja u njima. Može se koristiti kao indikator stanja pojedinih dijelova nekog stroja. Pri tome radnik može postupiti odgovarajuće i u stvarnom vremenu pratiti rezultate. Dopunjena stvarnost se koristi i pri upravljanju proizvodnih procesa, tako da se različite strane vizualno mogu obavijestiti o promjenama na određenim maketama uređaja ili AR-grafa.



Slika 1.16 Vizualizacija modela cijevi (lijevo) i AR podrška kod popravka printera (desno)

2. Programsko rješenje tehničke demonstracije računalne igre

U sklopu praktičnog dijela završnog rada napravljeno je programsko rješenje tehničke demonstracije računalne igre. Ukratko ćemo proći kroz tehnologije koje smo upotrijebili, te određene faze razvoja.

Programska implementacija predstavlja jednostavnu igru za dva igrača – auto nogomet. Sama igra je više tehnička demonstracija nego proizvod spreman za tržište. Cilj svakog igrača u igri je da lopticu na centru igrališta odgura u suigračev gol. Predefinirana logika nakon 5 minute odredi pobjednika s najviše golova, ili prijevremeno proglasi pobjednikom onog igrača koji prvi ostvari 5 golova. Vozilima se upravlja tipkovnicom (plavi igrač – strelice i tipka Enter za kočnicu, te crveni igrač – tipke WASD i Space)

2.1. Korištene tehnologije

U svijetu se koriste mnogi alati i tehnologije za razvoj dopunjene stvarnosti, ali ćemo ovdje nabrojati i ukratko opisati neke koje smo koristili za razvoj tehničke demonstracije.

Za programsku implementaciju je korišten programski jezik C# 3.0 pomoću Microsoft Visual C# 2008 Express Edition. Sva iscrtavanja su napravljena pomoću Microsoft XNA Game Studio 3.1 kojim se služila platforma za dopunjenu stvarnost Goblin XNA.

2.1.1. Razvojno okruženje Microsoft XNA

Microsoft XNA je niz alata (Framework) koji služe za izgradnju i upravljanje interaktivnim video igrama. Napravljen je s ciljem da se programer oslobodi pisanja koda koji se ponavlja. XNA se javno pojavio 2006. godine, a 3.1 verzija, koju koristimo u programskom ostvarenju, je izašla u lipnju 2009. godine. Zadnja verzija, XNA 4.0 se pojavila u rujnu 2010. godine, međutim nije korištena zbog promjena u kompajleru, kao i nemogućnosti sinkronizacije s Goblin XNA 3.5. Svi objekti koji se mogu stvoriti u XNA-u već u sebi sadrže predefiniran kod koji onda programeri upotpunjavaju svojim idejama. Video igre u XNA-u mogu biti pisane u bilo kojem programskom jeziku koji podržava .NET Framework, ali je službeno jezik za XNA C#. XNA osim iscrtavanja ima mnogo više mogućnosti poput podrške za zvuk ili mrežnu komunikaciju te je kompatibilan s raznim fizikalnim i grafičkim pogonima.

2.1.2. Platforma Goblin XNA

Goblin XNA je platforma za razvoj 3D površina, uključujući površina dopunjene kao i virtualne stvarnosti, s naglaskom na razvoj igara. Goblin XNA se koristi različitim tehnologijama da bi upotpunila svoju funkcionalnost – s obzirom na to može se proširiti sustavima za fiziku, zvuk, praćenje markera i slično.

Framework ima podršku za graf scene sličan OpenSG-u, 6DOF praćenje pozicije i orijentacije pomoću ALVAR i ARTag sustava za praćenje, 3DOF praćenje pomoću GPS-a, hvatanje videa pomoću DirectShow ili PGRFly tehnologije, fiziku pomoću niza pokretača, mrežu pomoću Lindgrena, kao i podršku za 2D GUI sustav.

Sučelje Goblin XNA sadrži 10 čvorova scene: Geometry, Transform, Light, Camera, Particle, Marker, Tracker, Sound, LOD (Level of Detail), Switch. Ima striktnu hijerarhiju – definira stablo grafa scene počevši od korijena (Root). Pomoću pojedinih čvorova možemo definirati različite elemente scene – jednostavne objekte, transformacije nad tim objektima, svjetlost, kameru i slično.

Goblin XNA sadrži i niz alata koji omogućuju samostalno kreiranje markera za pojedini marker sustav, kalibraciju kamere i grafički prikaz stabla grafa scene.

2.1.3. Fizikalni pokretač Newton Game Dynamics

Newton Game Dynamics je open-source solucija pod zlib-licencom za simulaciju fizike u stvarnom vremenu. Pokretač (*engine*) su razvili Julio Jerez i Alain Suero. Deterministički rješava probleme detekcije sudara i ostalih dinamičkih ponašanja.

Ovaj pokretač se koristi za simulaciju cijele fizike u tehničkoj demonstraciji (gravitacija, kolizije, interakcija, masa i slično).

2.1.4. Marker sustav ALVAR

Da bi demonstrirali tehnologiju dopunjene stvarnosti koristit će se ALVAR marker sustav. ALVAR je digitalno generiran, znanstveno verificiran i pouzdan marker sustav za dopunjenu stvarnost. To je programska podrška za računalni vid koja koristi markere da uskladi stvarne i virtualne kamere. ALVAR prepoznaje posebne crno-bijele kvadratične markere, pronalazi kut pod kojim stoji, i onda postavi matricu za prikaz objekta tako da se kasnije operacije ostvarivanja prikaza pojavljuju relativno u odnosu na polje, i stoga, relativno u odnosu na stvarni 3D svijet. Drugim riječima, ALVAR markeri dozvoljavaju programskoj podršci da kalkulira gdje će ubaciti virtualne elemente tako da se prikladno prikažu na proširenoj slici. Da bi se koristio ALVAR, uzorci markera se moraju isprintati i postaviti na mjesta gdje bi korisnik htio da se pojave virtualni elementi.

ALVAR ima SDK koji uključuje biblioteku markera, programsku podršku da ih se identificira u slikama, kao i kod koji pomaže s učitavanjem i prikazom 3D modela. Mi ćemo koristiti poveznik za Goblin XNA koji će nam omogućiti da se služimo ALVAR sustavom za praćenje.

2.1.5. Biblioteka funkcija Open Computer Vision

Za prevođenje ALVAR poveznika s XNA sučeljem korišten je Visual C++ Express Edition, kao i OpenCV biblioteka. OpenCV je open source biblioteka za računalni vid s fokusom na aplikacije u stvarnom vremenu. Sadrži preko 500 funkcija namijenjenih za razna područja računalnog vida.

2.1.6. Biblioteka funkcija DirectShow.NET

DirectShow.NET mogućuje potpuniji pristup Microsoftovoj DirectShow funkcionalnosti iz .NET aplikacija. Ona je povezana s video streamanjem i potrebna za potpunu funkcionalnost kamere. Prema developeru, Microsoftovo rješenje za pristup DirectShowu iz .NET-a nije ni približno cjelovito poput sučelja za C++. DirectShow.NET dozvoljava ALVAR sustavu za praćenje da preuzme informacije od web kamere i interpretira ih tako da Goblin XNA može odgovarajuće postupiti.

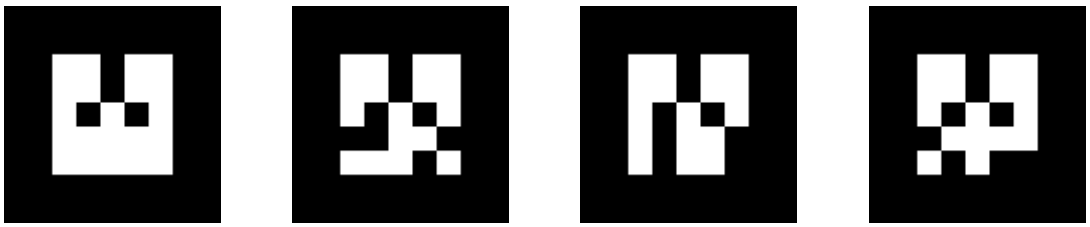
2.2. Kalibracija kamere i izrada markera

Unutar paketa Goblin XNA dolaze dva alata koja bi trebala pomoći pri prilagođavanju sustava našim vlastitim potrebama.

Prvi, CameraCalibration, je zadužen za kalibraciju naše vlastite kamere koju koristimo za praćenje markera. Program obrađuje 50 slika probnog markera i na osnovu toga izrađuje kalibracijsku XML datoteku koju koristimo u našim projektima.

Drugi alat, MarkerLayout, nam pomaže kod izrade vlastitih markera za ALVAR sustav za praćenje. Programu specificiramo sliku markera u PNG formatu, nakon čega on izradi TXT datoteku s opisom markera.

Postoje i drugi načini za kalibraciju kamere i izradu markera, ali se ovaj ispostavio najpraktičniji u kombinaciji s Goblin XNA platformom.



2.3. Implementacija i razvoj rješenja

Razvoj je krenuo s postavljanjem samog frameworka kako bi zadovoljio naše potrebe. Zbog toga smo kreirali klasu ZavrsniRad koja je trebala sadržavati glavne metode za njegov ispravan rad.

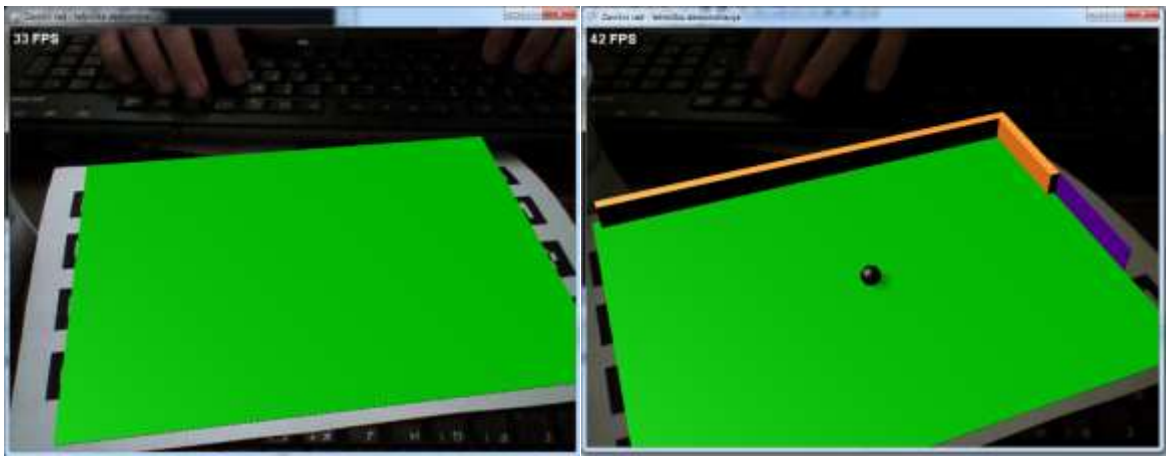
Tehnička demonstracija u ovom radu je svedena na minimalističke elemente kako bi se prikazale neke osnove grafike, rada s okruženjem i dizajna programske podrške.

Platforma dopunjene stvarnosti nam u ovom primjeru omogućuje jednostavnu promjenu kuta gledanja na naše igralište i objekte na njemu. Još jedna karakteristika AR-a dolazi do izražaja, naime samo igralište na kojem se odvija radnja se može staviti u različita stvarna okruženja koja mogu vizualno nadopuniti scenu – na taj način uljepšati ambijent (Slika 2.6).

Kao što to inače biva [13], implementacija igre je prošla kroz nekoliko faza:

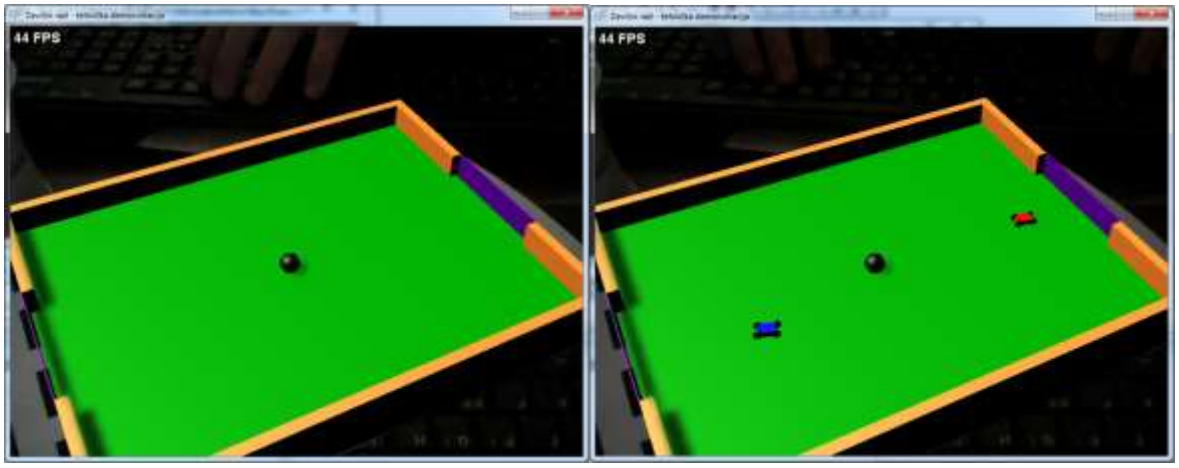
1. Prilagodba okruženja za praćenje markera
2. Stvaranje objekata i njihov prikaz
3. Definiranje logike i korisničkog sučelja

Kad je praćenje markera postavljeno, stvorena je površina za igranje, te se ubrzo nakon toga krenulo s pozicioniranjem prvih objekata (Slika 2.1).



Slika 2.1 Kalibriranje praćenja scene i pozicioniranje prvih objekata

Nakon programske izrade pojedinih objekata, oprezno su pozicionirani na scenu. Pri tome se vodilo računa o geometrijskom rasporedu u prostoru, tako da objekti budu u skladu (Slika 2.2).

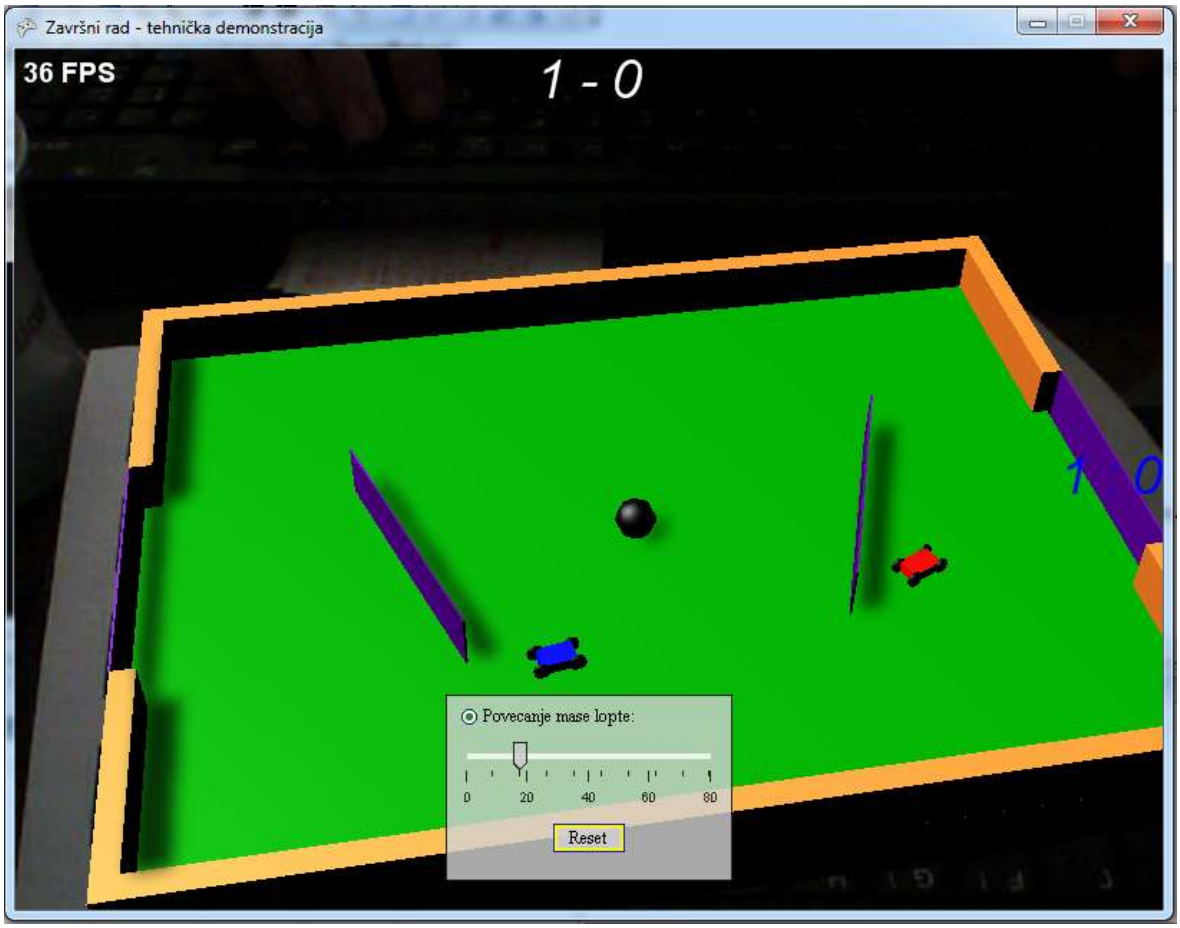


Slika 2.2 Igra poprima oblik alfa verzije

Definirana su pravila igre i izveden jednostavno grafičko sučelje. Grafičko sučelje je pokazivalo stanje rezultata, vremena, te ponudilo mogućnost da se dinamički promjeni jedan uvjet – poveća masa kuglice. Sučelje ima i tipku „reset“ koja igrače i sve uvjete resetira na početne vrijednosti.

Nakon dosta testiranja alfa verzije programa napravljene su neke promjene kako bi se utjecalo na igrivost. Izgrađene su dvije prepreke koje bi igru trebale učiniti dinamičnijom. Iz toga je proizašla trenutna verzija tehničke demonstracije (Slika 2.3).

U zadnjoj verziji je programski kod proširen mogućnošću Goblin XNA sučelja da grafički prikaže izgled grafa scene. Ta mogućnost je skrivena u izvršnoj verziji aplikacije, ali se može pokrenuti jednostavnim odkomentiranjem tog dijela koda.



Slika 2.3 Konačni verzija

2.3.1. Opis okruženja

Prije postavljanja okoline za rad potrebno je prvo razumjeti životni vijek jedne XNA igre, odnosno jedne Goblin XNA igre. Pri stvaranju novog projekta, uz konstruktor, sam framework deklarira pet klasa:

1. `Initialize()`
2. `LoadContent()`
3. `UnloadContent()`
4. `Update()`
5. `Draw()`

Prije samog konstruktora u kodu se deklariraju dvije klase: `GraphicsDeviceManager` i `SpriteBatch`. Prva upravlja samim grafičkim uređajem, a druga 2D grafikom unutar aplikacije. U glavnom konstruktoru imamo instanciran `GraphicsDeviceManager` i postavljeno jedno novo svojstvo (gdje se vrijednost `"Content"` odnosi na poddirektorij solucije koja sadrži dodatke za projekt).

```
public ZavrsniRad()
{
    grafika = new GraphicsDeviceManager(this);
    Content.RootDirectory = "Content";

}
```

U `Initialize()` metodu obično stavljamo negrafički kod za inicijalizaciju aplikacije. Također, tu pozivamo `base.Initialize()`, čime se inicijaliziraju druge komponente, poput grafičkog uređaja i `LoadContent()` metode (učitava sadržaj).

Glavna petlja igre se sastoji od naizmjeničnog pozivanja `Update()` i `Draw()` metoda, te igra petlju vrti dok joj se eksplicitno ne kaže da izađe iz nje. Nakon toga se poziva `UnloadContent()` i igra završava.

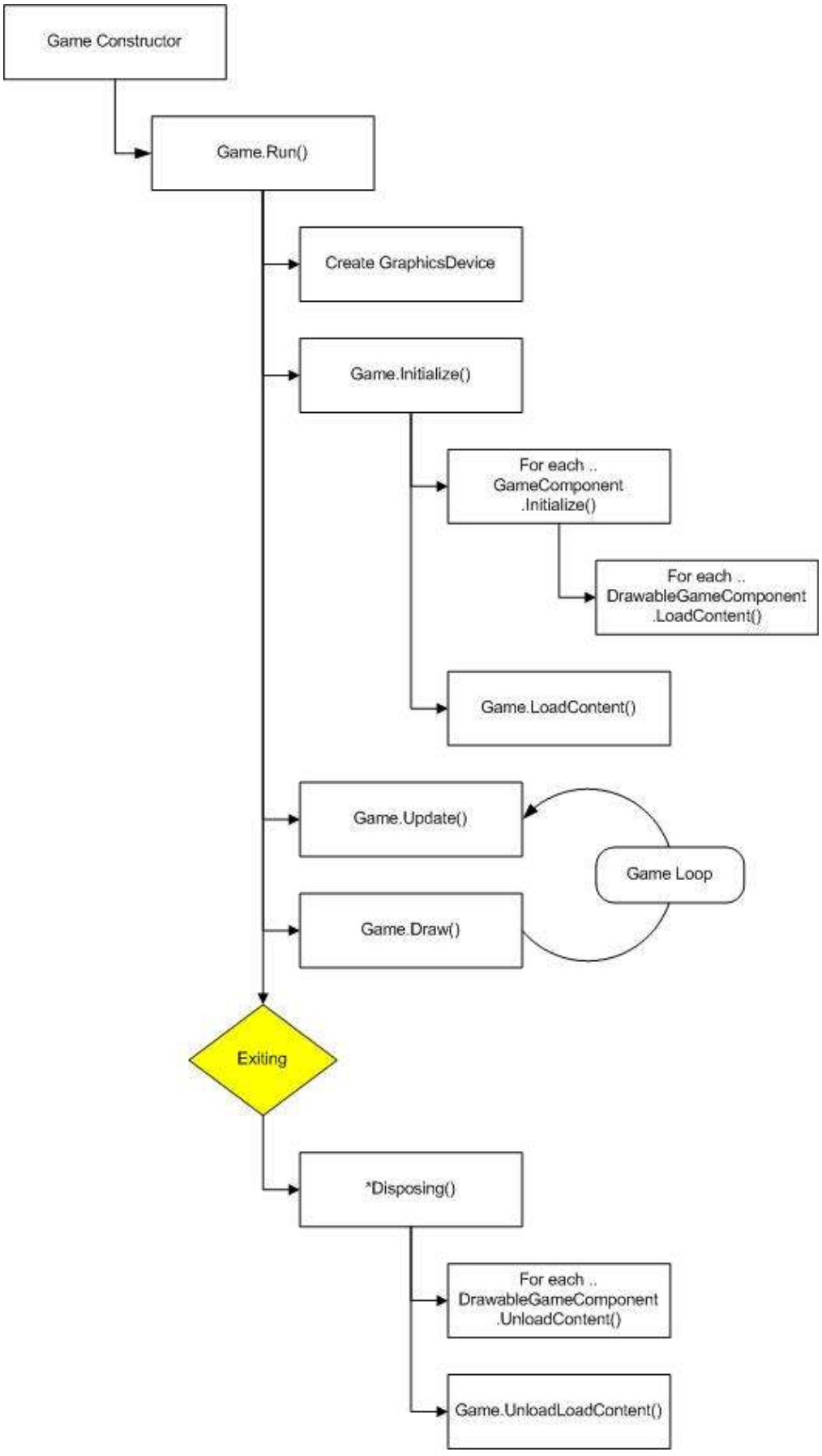
Prvo smo stvorili dva objekta, graf scene i font koji koristimo. Graf scene definira logičku strukturu scene, koja se po konvenciji reprezentira kao hijerarhijski aciklički graf od čvorova grafa scene.

```
Scene scena;
MarkerNode markerNode;

SpriteFont textFont;
```

`SpriteFont` se može referencirati s `UI2Renderer` objektom, te puni se u `LoadContent()` metodi, gdje je `"Sample"` font u `Content` direktoriju:

```
textFont = Content.Load<SpriteFont>("Sample");
```



Slika 2.4 Životni vijek XNA igre

2.3.2. Oblikovanje elemenata projekta

Kako bi bolje organizirali elemente scene stvorili smo neke pomoćne metode koje se inicijaliziraju na početku izvođenja. To su metoda za konfiguriranje praćenja markera (`SetupMarkerTracking()`), metoda za postavljanje svjetlosti (`CreateLights()`), metoda za stvaranje površine (`CreateGround()`), metoda za stvaranje objekata (`CreateObjects()`), te metoda za izradu 2D GUI-a (`Create2DGUI()`).

U prvom redu stvaramo uređaj za snimanje koji koristi DirectShow biblioteku. Različite kombinacije rezolucije i frame ratea su dozvoljeni u ovisnosti o uređaju.

Nakon toga još postavimo sustav za praćenje koji koristi ALVAR biblioteku, kažemo mu da se koristi za našu scenu, te prikažemo sliku kamere u podlozi. Za potrebe testiranja moguće je upotrijebiti dvije kamere – jednu za praćenje markera, i drugu za prijenos slike na ekran.

```
private void SetupMarkerTracking()
{
    DirectShowCapture captureDevice = new DirectShowCapture();
    captureDevice.InitVideoCapture(0,
    FrameRate._60Hz, Resolution._640x480, ImageFormat.R8G8B8_24,
    false);
    scena.AddVideoCaptureDevice(captureDevice);

    IMarkerTracker tracker = new ALVARMarkerTracker();
    ((ALVARMarkerTracker)tracker).MaxMarkerError = 0.02f;
    tracker.InitTracker(captureDevice.Width,
    captureDevice.Height, "calib.xml", 9.0);

    scena.MarkerTracker = tracker;

    scena.ShowCameraImage = true;
}
```

Naša igra se sastoji od jednostavnih objekata/modela koji su definirani unutar Goblin XNA platforme. Uglavnom su se koristili kvadri, i sfera za loptu. Nad tim objektima izvodimo različite transformacije kako bi ih oblikovali. Slijedi odsječak koda u kojem na primjeru sfere (lopta iz igre) demonstriramo neka obilježja:

```
private void CreateObjects()
{
    // Lopta
    loptaNode = new GeometryNode("Lopta");
    loptaNode.Model = new Sphere(1.8f, 8, 8);
    loptaNode.Physics.Collidable = true;
    loptaNode.Physics.Interactable = true;
    loptaNode.Physics.ApplyGravity = true;
    loptaNode.Physics.Mass = MASA_KUGLICE;
    loptaNode.AddToPhysicsEngine = true;
    loptaNode.Physics.Shape = ShapeType.Sphere;
    loptaNode.Model.CastShadows = true;
    loptaNode.Model.ReceiveShadows = true;

    loptaTransNode = new TransformNode();
    loptaTransNode.Translation = new Vector3(0, 0, 0.1f);

    Material loptaMaterial = new Material();
    loptaMaterial.Diffuse = Color.Black.ToVector4();
    loptaMaterial.Specular = Color.White.ToVector4();
    loptaMaterial.SpecularPower = 10;
    loptaNode.Material = loptaMaterial;

    markerNode.AddChild(loptaTransNode);
    loptaTransNode.AddChild(loptaNode);

}
```

Inicijalizacijom `Sphere` objekta mu dajemo tri argumenta: radijus, broj linija dužine oko uspravne y-osi, broj linija širine okomitih na y-os. Dva zadnja argumenta definiraju rezoluciju sfere. Što je veći broj, rezultirajuća sfera je bliža stvarnoj aproksimaciji sfere (i više se poligona koristi za tu aproksimaciju).

Dalje definiramo fizičke karakteristike objekta koje nam omogućuje Newton Dynamics (mogućnost sudaranja, reakcije na vanjske utjecaje, gravitacija i masa).

Inicijalno je objekt definiran da bude pozicioniran u našeg centar koordinatnog sustava (u ovom slučaju u odnosu na marker). Ako želim promijeniti njegovu lokaciju (translacija), ili možda veličinu (skaliranje) ili orijentaciju (rotacija), to radimo s geometrijskim transformacijama koje sistematski mijenjaju te vrijednosti. Na primjeru translacija je prikazana kao vektor tri decimalne vrijednosti u programskoj strukturi XNA platforme - Vector3.

Nakon toga stvaramo materijal koji nanosimo na objekt, te nam samo još ostaje ubacivanje objekta u graf scene. U ovom primjeru dodajemo loptu kao dijete čvoru za njenu transformaciju, tako da ona na nju utječe. Transformacija lopte se onda dodaje korijenskom čvoru kao dijete.

Na sličan način se definira i površina markera u `CreateGround()` metodi, s tim da u njemu definiramo korijenski čvor.

```
markerNode = new MarkerNode(scena.MarkerTracker, "ALVARGroundArray.xml");

scena.RootNode.AddChild(markerNode);
```

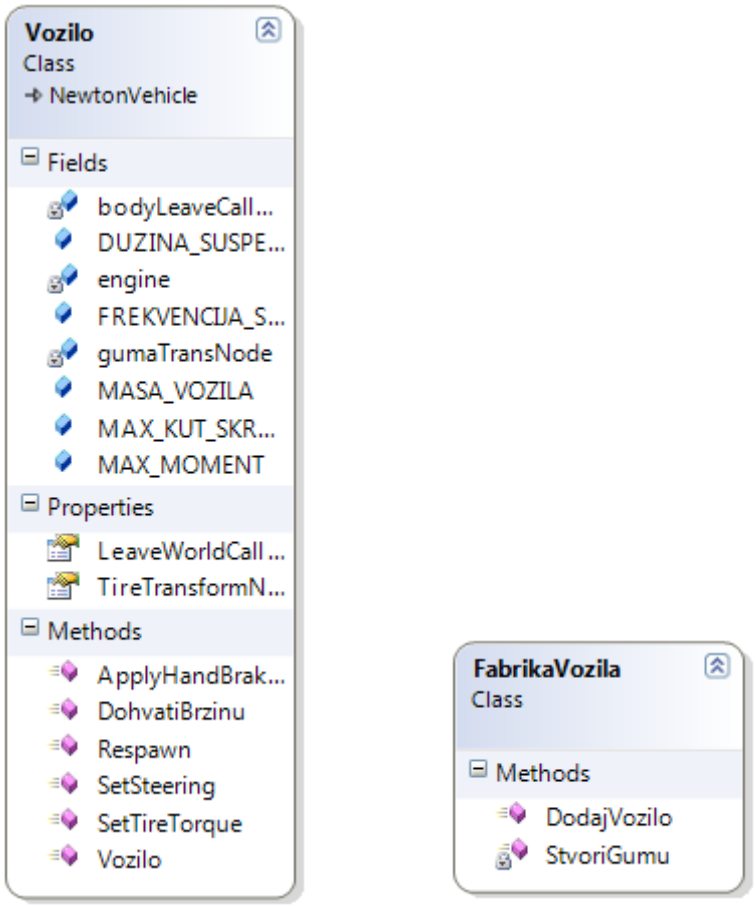
Ista logika vrijedi za stvaranje izvora svjetlosti. `LightNode` objekt nam omogućuje da objesimo naš izvor svjetlosti na drvo čvorova koje čini naš graf scene. Ako je `LightNode` globalan, onda `LightSource` dodan na taj čvor može utjecati na sve objekte na sceni. S druge strane, ako je lokalan, utječe samo na braću i potomke tog čvora.

```
        private void CreateLights()
        {
            // Stvori direkcijski izvor svjetlosti
            LightSource lightSource = new LightSource();
            lightSource.Direction = new Vector3(1, -1, -1);
            lightSource.Diffuse = Color.White.ToVector4();
            lightSource.Specular = new Vector4(0.61f, 0.61f, 0.61f, 1);

            // Stvori čvor svjetlosti koji će držati izvor
            LightNode lightNode = new LightNode();
            lightNode.LightSource = lightSource;

            scena.RootNode.AddChild(lightNode);
        }
```

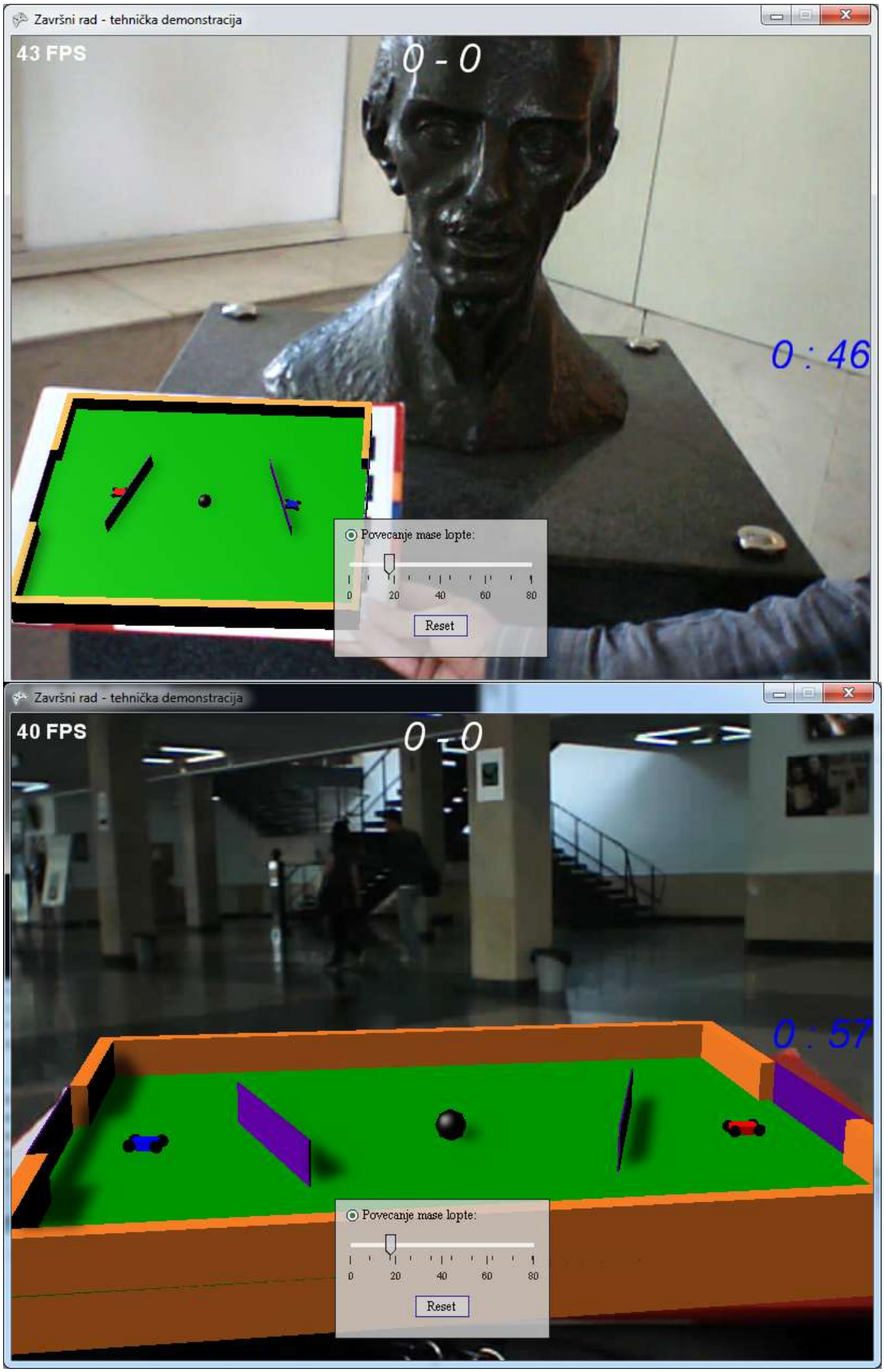
Vozila su izgrađena na osnovu NewtonVehicle klase vozila implementirane u programskom jeziku C++, te su korištene poveznice platforme Goblin XNA s Newton fizičkim sučeljem za njihovo ostvarenje. Klasa Vozilo opisuje objekt, dok FabrikaVozila služi kao pomoćna klasa za dodavanje vozila na scenu.



Slika 2.5 Dijagram razreda Vozilo bi FabrikaVozila

Grafičko 2D sučelje je formirano korištenjem objekta `G2DPanel` koji postoji unutar Goblin XNA platforme. Na njega su postavljeni klizač za dinamičko mijenjanje mase loptice i gumb za reset cijele igre.

Logika igre se nalazi u `Update()` metodi, i sastoji se od brojača vremena i pojedinih uvjete za ostvarenje pobjede. Ti uvjeti se provjeravaju svaki „otkucaj“ internog sata XNA frameworka.



Slika 2.6 „Auto nogomet“ u različitim stvarnim okruženjima

Zaključak

Izrada igre na kraju se pokazala kao poprilično velik posao. Iako je rađena samo s osnovnim grafičkim elementima, pokazalo se da dopunjena stvarnost predstavlja dodatni izazov u sinkronizaciji pojedinih elemenata.

Određene su stvari mogle biti drugačije izvedene, ali trenutna struktura rada je proširiva, tako da se u budućim verzijama može jednostavno nadograditi. U sklopu potencijalnih kasnijih verzija moguće je prekrajanje idejnog okruženja i dodavanje dodatnih funkcionalnosti, kao što su, primjerice:

- Estetski ljepši i pristupačniji dizajn elemenata i sučelja
- Ručna kontrola uvjeta pobjede i karakteristika igrača i prepreka
- Podrška za do četiri igrača i više različitih površina za igranje
- Manipulacija gravitacijskog polja same površine
- Dinamički elementi koji mijenjaju stanje lopte i igrača (engl. *powerups*)
- Mogućnost upravljanja vozila s markerima

Konačni rezultat je bio zadovoljavajući, no za razvoj potpunog proizvoda potrebno je dugoročnije testiranje i modifikacija. Pored običnih problema razvoja programske potpore, postoje i problemi samih uvjeta pod kojima funkcionira dopunjena stvarnost (osvjetljenje, udaljenost markera, fizička kvaliteta kamere, programska podrška za kameru). Goblin XNA je ponudio relativno dobre rezultate za ovu primjenu, te će se s budućim verzijama sigurno samo poboljšati.

Mobiteli, tableti i igraće konzole predstavljaju samo neke od medija nove generacije na kojima dopunjena stvarnost prosperira, te budi zainteresiranost komercijalnog tržišta. Opipljiva sučelja poput ovih su budućnost svijeta računala i dok se dalje razvijaju jednom ćemo biti u mogućnosti ih iskoristiti do njihovog punog potencijala, i stvoriti istinski intuitivne, štoviše, „čarobne“ aplikacije koje će promijeniti svijet.

(Adnan Abdagić)

Literatura

[1] Milgram, P., Takemura, H., Utsumi, A., Kishino, F. Augmented Reality: A class of displays on the reality-virtuality continuum

[2] Cawood, S., Fiala, M. Augmented Reality: A Practical Guide

[3] Wagner, D., Schmalstieg, D. Graz University of Technology. Making Augmented Reality Practical on Mobile Phones

[4] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. Recent Advances in Augmented Reality

[5] Rosenblum, L. Virtual and Augmented Reality 2020.

[6] Sutherland, I. E. The University of Utah. A Head-Mounted Three Dimensional Display

[7] Sutherland, I. E., The Ultimate Display

[8] Vögele, M., Augmented Reality: Projectors

[9] Oda, O., Lister, L. J., White, S., Feiner, S. Department of Computer Science, Columbia University. Developing an Augmented Reality Racing Game

[10] Oda, O., Feiner, S. Rolling and Shooting: Two Augmented Reality Games

[11] Feiner, S. Augmented reality: A new way of seeing. Scientific American, 286(4), 2002, 34–41.

[12] Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P., and Piekarski, W. First person indoor/outdoor augmented reality application: ARQuake. Personal and Ubiquitous Computing, 6(1), 2002, 75–86.

[13] Truesdell, J. GameBlocks: DM Masters Project Design Document, 2010.

[14] Whitaker, R. T., Crampto, C., Breen, D. E., Tuceryan, M., Rose, E., Object Calibration for Augmented Reality. ECRC

Sažetak

RAZVOJ IGARA U OKRUŽENJU DOPUNJENE STVARNOSTI

Dopunjena stvarnost predstavlja pojam koji je danas sve u prisutniji u raznim sferama života (edukacija, arhitektura, zabava i sl.). Jedan od najaktualnijih predstavnika je industrija igara – ljudi teže da prošire svoj doživljaj virtualnog svijeta, te zbog toga žele virtualne objekte dovesti u stvarnost. Ovdje se koristi *pinhole* model kamere za grafičko predstavljanje i obradu unutar računala. Za miješanje slike je korišteno video miješanje (ekran računala i web kamera). Implementacija programskog rješenja (igra „auto nogomet“) je napravljena u jeziku C#, uz pomoć različitih tehnologija – za ostvarivanje praćenja markera (ALVAR), simulaciju fizike (Newton Dynamics), računalnog vida (OpenCV) – koje je platforma Goblin XNA ukomponirala u jednu cjelinu.

Ključne riječi: dopunjena stvarnost, razvoj igara, XNA

Abstract

GAME DEVELOPMENT IN AN AUGMENTED REALITY ENVIRONMENT

Augmented reality is a term ever more present in various aspects of modern life. (e.g. education, architecture, entertainment, etc.) One of it's most prominent examples is the game industry - people strive to expand their virtual world experience by bringing virtual objects into a real-life environment. A pinhole camera model is used for graphical representation and computer processing. Image mixing is realized through video mixing (computer screen and a web camera). The program solution ("auto-soccer" game) is implemented using the C# language through various technologies - marker motion tracking (ALVAR), physics simulation (Newton Dynamics), computer vision (OpenCV) - all incorporated into a single entity using the Goblin XNA platform.

Keywords: augmented reality, game development, XNA

.

.

.