UDC: 004.92:004.43 Info M: str. 14-25

RAZVOJ APLIKACIJE SA 3D INTERAKCIJOM: KINECTCITY
DEVELOPMENT OF AN APPLICATION WITH 3D INTERACTION: KINECTCITY

Dario Mirović, Dragan Okanović, Jovan Radivojša, Nemanja Lučić i Đorđe Đurđević
Univerzitet u Beogradu - Elektrotehnički fakultet
LARGO Laboratorija za Računarsku grafiku i virtuelnu realnost

REZIME: Interakcija između čoveka i računara je disciplina koja ima za cilj da prouči i definiše bezbedne, intuitivne i fleksibilne načine na koje čovek računaru zadaje komande, a računar izveštava o rezultatima obrade. 3D interakcija je posebna vrsta interakcije kod koje je od značaja pozicija (u 3D prostoru) objekata posredstvom kojih se vrši interakcija. Tradicionalno, korisnik računaru svoje zahteve saopštava sekvencom odgovarajućih komandi zadatih putem ulaznih uređaja, kao što su miš i tastatura, a računar korisniku prikazuje efekte primene zadatih komandi na izlaznom uređaju, poput monitora. Napredovanje tehnologije dovelo je do pojave komercijalno raspoloživih jeftinih uređaja koji mogu da detektuju i prate ljudsko telo u prostoru. Time je otvorena mogućnost da se ostvari 3D interakcija sa računarom, čak i u skromnim, svakodnevnim uslovima. U ovom radu predstavljen je razvoj 3D aplikacije KinectCity, u kojoj korisnik interakciju ostvaruje pokretima tela i glasom. Aplikacija je razvijena upotrebom alata Unity3D, a za detekciju i praćenje ljudskog tela korišćen je uređaj Kinect. U prisustvu odgovarajućeg hardvera, aplikacija može da formira stereoskopski prikaz, što značajno pojačava iskustvo 3D interakcije sa virtuelnim svetom.

KLJUČNE REČI: 3D interakcija, virtuelna realnost, stereovizija, 3D grafika, Unity3D, Kinect.

ABSTRACT: Human-computer interaction is a discipline intended to examine and define safe, intuitive and flexible means for human users to issue instructions to computers, and for computers to inform about outcomes. 3D interaction is a special kind of interaction where the position (in 3D space) of objects involved in interaction is of great importance. Traditionally, the user issues his/her requests through a sequence of commands on input devices, such as mouse or keyboard, and the computer displays the results on an output device, such as monitor. Recent advances in technology resulted in wide availability of inexpensive devices that can detect and track human body in 3D space. This opens a possibility to establish 3D interaction with computers, even in casual conditions. This paper presents the development of a 3D application KinectCity, in which the user establishes interaction with body gestures and spoken commands. The application was developed in Unity3D. Kinect was used for body detection and tracking. In presence of adequate hardware, the application can present stereoscopic viewing, which significantly increases the 3D interaction experience with the virutal world.

KEY WORDS: 3D interaction, virtual reality, stereovision, 3D graphics, Unity3D, Kinect

1. UVOD

Kroz interakciju sa računarom, korisnik uspostavlja komunikaciju, najčešće neverbalnu, kojom računaru saopštava svoje zahteve, a od računara dobija povratnu informaciju o rezultatu. Od samog početka razvoja računarske tehnike, uočen je problem uspostavljanja efikasnog i fleksibilnog načina zadavanja komandi računaru. Tastatura se vrlo brzo nametnula kao opšteprihvaćen ulazni uređaj i taj status je zadržala do današnjih dana, sa tendencijom da tako i ostane još duži niz godina. Ipak, za mnoge primene tastatura nije dovoljno efikasno sredstvo, naročito u domenu računarske grafike i vizuelizacije. Sa pojavom prikazivača sposobnih da prikažu sliku sintetizovanu upotrebom geometrijskih primitiva, a ne isključivo teksta, pojavila se potreba korisnika da direktno (na ekranu) pokaže predmet njegove predstojeće komande. To je dalje dovelo do pojave specijalizovanih uređaja poput svetlosne olovke ili miša, a kasnije i do ekrana osetljivih na dodir. Nesumnjivo, takvi uređaji su značajno pojednostavili upotrebu računara i učinili ih pristupačnijim prosečnom korisniku.

Razvoj tehnologija ulaznih i izlaznih uređaja računara, koji su u osnovi interakcije između korisnika i računara, doveo je do komercijalne raspoloživosti uređaja koji su do nedavno bili samo predmet maštanja i tema naučno-fantastičnih filmova. *Glass* [1], proizvod kompanije Google, predstavlja jedan od prvih komercijalnih uređaja u tehnologiji podesnoj za nošenje (eng. wearable technology). U pitanju je uređaj koji sadrži minijaturnu kameru i prikazivač, predviđen da se učvrsti za ram naočara. Može da se poveže bežičnim putem sa Internetom, a korisnik komanduje glasom. Kompanija Sony je sredinom marta 2014. godine objavila da radi na razvoju sistema za virtuelnu realnost [2] koji namerava da upotrebi kao dodatak svojoj popularnoj konzoli za igre Playstation 4. U pitanju je uređaj koji korisnik nosi na glavi, preko očiju, opremljen senzorima za praćenje pokreta glave. Uređaj sadrži ekran visoke rezolucije koji zauzima najveći deo vidnog polja, tako da korisnik ima pojačan osećaj da se nalazi "uvučen" u virtuelni svet. Svako oko prima delimično drugačiju sliku, čime se postiže realističan stereoskopski efekat. Slično kompaniji Sony, kompanija Oculus VR razvija Rift [3], svoju verziju uređaja za virtuelnu realnost, u saradnji sa vodećim svetskim kompanijama za razvoj i proizvodnju video-igara. Za razliku od kompanije Sony, koja je svoj proizvod orijentisala ka svojim konzolama za igru, Rift je namenjen opštoj upotrebi i tako je privukao značajnu pažnju javnosti kada je 2013. godine ponudio zainteresovanim kompanijama i entuzijastama da nabave ranu razvojnu verziju uređaja i otpočnu razvoj softvera. Pored navedenih sistema koji su prvenstveno namenjeni personalnoj upotrebi, postoje i CAVE (Cave Automatic Virtual Environment) sistemi [4] koji se sve više koriste na kolaborativnim projektima, prilikom planiranja ili testiranja prototipova pre njihove materijalizacije. Kod ovih sistema, korisnik se nalazi u prostoriji oblika kocke

na čije zidove (od 3 do 6 strana) se projektuje slika virtuelnog prostora.

U pogledu ulaznih uređaja, u poslednjih nekoliko godina veliku pažnju javnosti privukli su uređaji sposobni da detektuju i prate pokrete ljudskog tela u prostoru. Leap Motion Controller [5] koncipiran je kao dopuna ostalim klasičnim ulaznim uređajima (tastatura, miš). Ima mogućnost precizne detekcije pokreta prstiju i šaka korisnika na relativno maloj distanci. Zbog toga ne može da prati celo ljudsko telo, a korisnik mora da bude u neposrednoj blizini računara da bi ga koristio. Kompanija Microsoft je 2012. godine na tržište plasirala uređaj Kinect [6]. Uređaj je opremljen kamerama za snimanje u vidljivom i infracrvenom (IC) delu spektra, kao i nizom mikrofona. Kroz softversku obradu ulaza uređaja, moguće je jednostavno detektovati i pratiti kretanje delova ljudskog tela kao i detektovati poziciju izvora zvuka u odnosu na uređaj. Unapređena verzija, pod imenom Kinect 2, koja se nedavno pojavila na tržištu, opremljena je kvalitetnijim senzorima povećane rezolucije, što omogućava veću preciznost prilikom detekcije oblika i pokreta. Kompanija Intel proizvela je RealSense [7], uređaj sličan uređaju Kinect po pitanju mogućnosti i namene. Za razliku od Kinect, koji predstavlja zaseban uređaj, predviđeno je da RealSense takođe dolazi integrisan u prenosive (*laptop* i *notebook*) računare.

Imajući u vidu nedavnu kupovinu kompanije Oculus VR od strane kompanije Facebook za približno dve milijarde dolara, konkurs na sumu od 1 milion dolara koji je raspisala kompanija Intel za razvoj napredne aplikacije koja koristi uređaj RealSense, kao i to da su u protekloj deceniji napravljene značajne investicije u razvoj uređaja za 3D interakciju i virtuelnu realnost, može se zaključiti da su vodeće svetske softverske i hardverske kompanije u sistemima za virtuelnu realnost prepoznale budućnost interakcije između čoveka i računara.

Prethodno navedene tehnologije su još uvek u razvoju i imaju određen broj tehnoloških nedostataka. Međutim, otklanjanje ovih nedostataka u budućnosti neće automatski rešiti i pitanje efikasnog, ergonomskog i intuitivnog komuniciranja sa računarom. Za korisnika može biti izuzetno frustrirajuće neprepoznavanje ili pogrešno prepoznavanje pokreta, naročito ako to dovede do neželjene aktivnosti poput brisanja podataka. Stoga je, paralelno sa razvojem tehnologija za 3D interakciju, neophodno razvijati i konkretne mehanizme za sprovođenje interakcije, prilagođene čak i korisnicima koji nemaju visok nivo obrazovanja u oblasti računarske tehnike, niti vremena da prolaze posebnu obuku da bi mogli da koriste računar primenom modernih sredstava za interakciju.

U ovom radu predstavljena je aplikacija KinectCity, razvijena u Laboratoriji za računarsku grafiku i virtuelnu realnost pri Elektrotehničkom fakultetu Univerziteta u Beogradu (http://largo.etf.rs), u kojoj korisnik, putem 3D interakcije, gradi i razgleda virtuelni grad. Za interakciju korišćen je uređaj Kinect. Korisnik zadaje svoje komande telesnim stavom i pokretima udova, a režime rada aplikacije menja glasovnim komandama, bez korišćenja miša ili tastature. Postoji i specijalan režim razgledanja vožnjom motocikla, kada korisnik naginjanjem tela kontroliše smer kretanja motocikla, a pružanjem ruku napred, odnosno povlačenjem ruku unazad, kontroliše brzinu. U radu su prikazana stečena iskustva u vezi sa razvojem sistema za 3D interakciju i prepoznavanja komandi zadatih pokretima tela. Aplikacija je osposobljena za prikazivanje virtuelnog sveta primenom stereovizije. Pri svakom crtanju scene, generišu se dve slike stereo para. Iako je aplikacija namenjena jednom korisniku, projektovanjem stereo para na platno omogućava se većem broju pasivnih posmatrača da prate aktivnosti korisnika.

Ostatak rada organizovan je na sledeći način. U drugom poglavlju ukratko je predstavljen uređaj Kinect i njegove mogućnosti, a zatim je opisana njegova upotreba u okviru aplikacije KinectCity. U trećem poglavlju data je funkcionalna specifikacija aplikacije, a najbitniji detalji njene implementacije dati su u četvrtom poglavlju. U zaključku su rezimirani rezultati rada i skicirane su smernice daljeg razvoja i istraživanja.

2. INTERAKCIJA UPOTREBOM UREĐAJA KINECT

Kinect [6] je linija proizvoda napravljenih od strane kompanije Microsoft i sastoji se od uređaja za detekciju pokreta (eng. motion sensing) koji su pre svega namenjeni Xbox 360 konzoli i omogućavaju korisniku da kontroliše aplikacije pokretima svog tela, bez potrebe za dodatnim klasičnim kontrolerima za igru. Prva generacija Kinect uređaja je predstavljena 2010. godine, dok je uređaj postao dostupan i na Windows platformi 2012. godine. Najnovija verzija Kinect uređaja je v2 i sa sobom je donela dodatne senzore i noviji SDK (eng. SDK – Software Development Kit) paket programa i alata za razvoj.

Kinect uređaj je sastavljen od nekoliko različitih senzora [8]: kamera za snimanje slike u boji, kamera za snimanje dubinske mape (eng. depth map), koja se sastoji od IC emitera i IC senzora, i četiri mikrofona (slika 1). Pored senzora, uređaj sadrži i motorizovani nosač pomoću kog je moguće softverskim putem podešavati nagib uređaja (senzora).

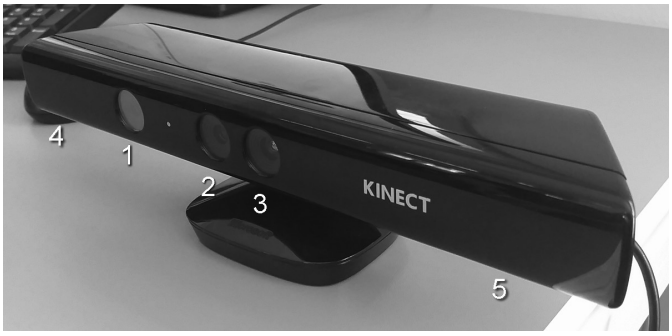
Uočavanje i praćenje objekata, kao i detekcija pokreta vrši se softverskim putem, na osnovu podataka prikupljenih sa senzora uređaja [9]. U SDK za Kinect, dostupno je gotovo rešenje za detekciju ljudskog tela i praćenje njegovog položaja u vidu skupa karakterističnih tačaka (na primer: levo rame, desni lakat) na osnovu kojih se jednostavno može rekonstruisati grub skelet. Detekcija delova skeleta vrši se u dve faze [10, 11]. U prvoj fazi se formira dubinska mapa koja sadrži rastojanja svakog elementa (piksela) mape od uređaja. U drugoj fazi se, iz dobijene mape, za svaki piksel određuje deo tela kojem dati piksel pripada. Na osnovu dobijenih rezultata računa se približna pozicija karakterističnih tačaka skeleta. Faze detekcije delova skeleta ilustrovane su na slici 2.

Prva faza se sastoji od emitovanja infracrvenih zraka prema unapred definisanoj šari (prostornom rasporedu) na objekte ispred senzora [10]. Na osnovu poznate šare, slike snimljene IC kamerom i unapred definisanog rastojanja između IC emitera i IC kamere, vrši se računanje približnog rastojanja (dubine) za svaki piksel pojedinačno. Zbog nepreciznosti samih senzora, kalkulacija koje se koriste i pretpostavki koje se uzimaju, može doći do raznih diskontinuiteta i nepravilnosti u rekonstruisanoj dubinskoj mapi. U praksi se ove pojave manifestuju naglim (skokovitim) pomerajima karakterističnih tačaka,

zbog čega se u obliku rekonstruisanog skeleta uočavaju razna neprirodna trzanja. U poglavlju 3 biće objašnjen način na koji je u ovom radu rešen uočen problem.

U drugoj fazi rekonstrukcije skeleta, za svaki piksel na dubinskoj mapi, procenjuje se deo tela na kojem se nalazi dati piksel. Za takvu klasifikaciju je korišćena šuma slučajnog odlučivanja (eng. randomized decision forest) [12]. Trening klasifikatora izvršen je od strane proizvođača velikim brojem stvarnih i sintetičkih slika [11]. Iako klasifikator u velikom broju slučaja daje željene rezultate, postoje situacije kada dolazi do problema prilikom prepoznavanja određenih položaja tela. Generalno problemi nastaju kada se delovi tela ili ne vide od strane senzora (zaklonjeni su drugim delovima tela, na primer ruka iza leđa) ili se međusobno preklapaju (odnosno na približno istoj su udaljenosti od senzora).

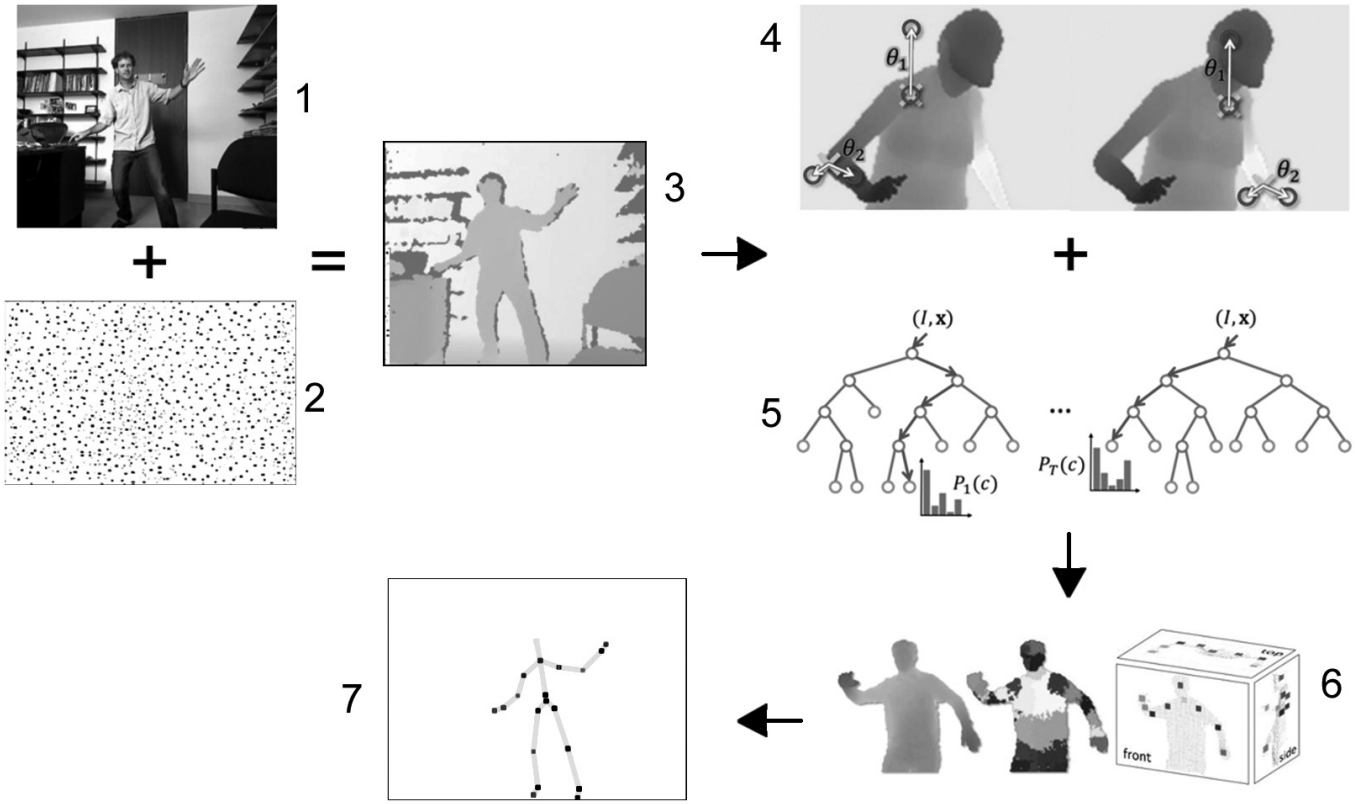
Iako ovakav način detekcije nije savršen u svakoj situaciji, njegova glavna prednost je u tome što je relativno jednostavan i savremene sisteme opterećuje u dovoljno malom obimu. Zahvaljujući tome, na savremenim sistemima su ostvarljivi rezultati pogodni za interaktivan rad (30 ili više slika u sekundi). Pored toga, pristup je pogodan za veliki broj različitih oblika (silueta) ljudskih tela koji postoje, što ga čini dovoljno dobrim za komercijalne svrhe kod kojih preciznost nije od suštinskog značaja. Takođe, cena procesa proizvodnje potrebnog hardvera i pratećeg softvera ostaje pristupačna velikom procentu korisnika.



Slika 1. – *Prednja strana uređaja Kinect (okrenuta ka korisniku). Emiter IC svetla (1) i IC senzor (3) služe za formiranje dubinske mape. Kolor kamera (2) daje konvencionalnu sliku u boji. Mikrofoni su raspoređeni sa donje strane uređaja (ne vide se na slici): jedan mikrofon na lokaciji (4), a tri mikrofona približno na lokaciji (5).*

Prilikom razvoja aplikacije koja je opisana u ovom radu, bilo je potrebno implementirati prepoznavanje gestura izvršenih različitim delovima tela koji bi se koristili za upravljanje aplikacijom. Zbog nedovoljne preciznosti uređaja Kinect za prepoznavanje gestura šaka (položaja zglobova i prstiju), u aplikaciji KinectCity gesture se izvode pokretima delova tela, najčešće levom i desnom šakom. Razmatrana su sledeća tri pristupa:

- **Klasifikacija putanje**: ova tehnika prati poziciju određene tačke (dela tela) u vremenu, u cilju opisivanja njene putanje nekom matematičkom funkcijom (analitičkom ili opi-



Slika 2. – *Faze konstrukcije skeleta. Posmatran korisnik (1) se osvetljava IC svetlošću prema šari (2), na osnovu čega se dobija dubinska mapa (3). Za svaki piksel dubinske mape vrši se računanje odlika datog piksela na osnovu nekoliko kriterijuma (4), a zatim se izračunate odlike koriste za klasifikaciju putem šume slučajnog odlučivanja (5) da bi se odgovarajući pikseli klasifikovali prema delu tela kojem pripadaju (6). Na osnovu toga vrši se konstrukcija izgleda skeleta (7). Delovi slike su preuzeti iz [10], uz odobrenje autora.*

sanom deo po deo) koja bi bila svrstana u neku od unapred definisanih kategorija. Primera radi, uspešna aproksimacija uzastopnih pozicija posmatrane tačke jednačinom prave bi navela na zaključak da je korisnik pokušao da izvede neki vid pravolinijskog pokreta u određenom pravcu i smeru. S druge strane, ako bi pozicije tačke približnije odgovarale kružnici, više nego nekoj pravoj liniji, zaključilo bi se da je korisnik imao nameru da izvrši kružni pokret.

- **Evaluacija sličnosti**: ova tehnika vrši klasifikaciju pokreta na osnovu toga koliko tačke praćene tokom vremena odstupaju od neke idealne putanje. Količine odstupanja za svaku tačku putanje pojedinačno se procenjuju i na osnovu prethodno izvršenog treninga vrši se selekcija kojoj gesturi bi najbolje odgovarao korisnikov pokret. Ova tehnika je poznata kao DTW (eng. dynamic time wrapping) algoritam [13].
- **Segmentacija pokreta**: ova tehnika je bazirana na utvrđivanju da li praćena tačka prolazi kroz unapred definisane segmente (regione, odnosno oblasti u 3D prostoru). U slučaju da praćena tačka prođe kroz sve segmente u ispravnoj sekvenci i u zadatom vremenskom intervalu, prepoznaje se određena gestura.

I pored toga što upotreba uređaja Kinect ima ozbiljne nedostatke, osećaj slobode pokreta prilikom upravljanja aplikacijom snažno podstiče potrebu za daljim istraživanjem ovakvog načina interagovanja. Razumno je očekivati da će buduće verzije ovog i sličnih uređaja tehnološki prevazići uočene probleme i time omogućiti još jednostavniji rad, kako korisnicima tako i programerima.

3. KINECTCITY

U ovom poglavlju predstavljena je aplikacija KinectCity i opisane su njene mogućnosti. Najpre je objašnjena namena aplikacije, a zatim su objašnjeni način detekcije korisnikovih komandi, način formiranja stereoskopskog prikaza, prikazivanje korisnikovog avatara, upravljanje 3D kursorom i režimi rada.

3.1 *Interaktivna izgradnja i razgledanje virtuelnog grada*

Aplikacija KinectCity napravljena je s ciljem izučavanja mogućnosti i tehnika 3D interakcije korisnika sa računarom. Interakcija korisnika sa ovom aplikacijom obuhvata pokrete delovima tela i glasovne komande. Za obe vrste interakcije korišćen je uređaj Kinect. U aplikaciji, korisnik se nalazi u ulozi graditelja grada. Na raspolaganju ima nekoliko kategorija građevina, od uličnih svetiljki i klupa do višespratnih solitera. U svakoj kategoriji postoji više različitih tipova objekata koje korisnik može da izabere i rasporedi u sceni. Izuzev izmena koje unosi korisnik, scena je u potpunosti statična. Osim uloge graditelja, korisnik ima mogućnost da razgleda izgrađen grad slobodnim kretanjem kroz 3D prostor upotrebom različitih režima kretanja koji se izvode pokretima delova tela.

Postoje dva režima rada, koji se pokreću iz glavnog menija. Prvi režim je učenje, odnosno uvežbavanje pravilnog izvođenja pokreta koje aplikacija može da prepozna, kao i uvežbavanje pravilnog zadavanja glasovnih komandi. Drugi režim je izgradnja odnosno razgledanje grada. Po ulasku u ovaj režim rada, korisnik bira jednu od ponuđenih scena u kojoj će da sprovodi svoje aktivnosti. Iako broj scena nije ograničen, kao dokaz o ispravnosti koncepta u trenutnoj verziji aplikacije realizovane su dve scene: dnevna i noćna (slike 3a i 3b). Dnevna scena se sastoji od brdovitog pošumljenog terena okruženog vodom (ostrvo), bez izgrađenih objekata. Noćna scena sadrži delimično izgrađen grad sa nekoliko ulica. Po izboru scene, korisnik bira mesto u sceni (terenu) gde će biti pozicioniran i odakle može da započne rad. Od ovog trenutka korisnik može da pređe jedan od tri podrežima rada aplikacije, u kojem spro-

a) b) c)

Slika 3. – *Primeri scena i interakcije u aplikaciji KinectCity. a) Dnevna scena sa otvorenom paletom za izbor kategorije objekata za izgradnju. b) Noćna scena, korisnik bira lokaciju na terenu odakle će započeti razgledanje. c) Interaktivna vožnja motocikla u dnevnoj sceni.*

vodi svoje aktivnosti: interaktivno razgledanje grada, interaktivna vožnja motocikla i interaktivna izgradnja, pri čemu se inicijalno nalazi u podrežimu razgledanja. Tekući podrežim menja se odgovarajućom gesturom. Režimi rada su detaljnije opisani u odeljku 3.6.

U podrežimu razgledanja, korisnik može slobodno da se kreće po sceni. Konkretan način kretanja bira se glasovnom komandom, a zatim se samo kretanje izvodi pokretima delova tela. Poseban način kretanja je vožnja motocikla. Iako se u tom režimu delimično gubi sloboda kretanja, pokazao se kao izuzetno zanimljiv i atraktivan za korisnika. U podrežimu izgradnje korisnik može da raspoređuje objekte na teren, bez mogućnosti promene svoje pozicije ili orijentacije. U ovom podrežimu poziciju desne šake korisnika prati 3D kursor (odeljak 3.5) kojim se "nose" objekti izabrani za raspoređivanje, po analogiji sa uobičajenom operacijom mišem *drag* u 2D grafičkim interfejsima. Pre postavljanja (izgradnje) objekata na teren, korisnik kroz poseban meni najpre bira kategoriju građevine, a zatim i konkretan tip građevine. Nakon toga, primerak odabranog tipa građevine desnom šakom postavlja na željena mesta na terenu, proizvoljan broj puta. Sve vreme tip građevine ostaje odabran za dalje postavljanje. Posebnom gesturom vrši se vraćanje na početak stanja postavljanja objekata na teren, odnosno prestanak postavljanja trenutno izabranog tipa objekta.

3.2 *Načini interakcije*

S obzirom na to da je tehnika segmentacije pokreta (videti poglavlje 2) jednostavna za implementaciju, a i uz relativno mali napor omogućava definisanje gesture proizvoljnih oblika, implementirana je u aplikaciji KinectCity.

U realizaciji izabranog sistema detekcije gestura, uvodi se pojam *aktivator*. Aktivator je određen deo tela koji može aktivirati određenu gesturu. Aktivatori se prate tokom vremena, što znači da se informacije o ranijim pozicijama aktivatora čuvaju za prethodnih 30-60 prikazanih slika. Jedan aktivator može biti vezan za više gestura, i obrnuto, jedna gestura može imati više aktivatora koji mogu napraviti pokret. Pod pokretom se podrazumeva definisanje svih regiona koji opisuju jedan pokret (gesturu), poredak regiona kroz koje aktivatori treba da prođu da bi aktivirali akciju gesture i maksimalno vreme za koje pokret mora biti izvršen da bi bio prepoznat. Jedan region se predstavlja pomoću jednog 3D oblika, koji može proizvoljno biti orijentisan u prostoru. Tokom razvoja pokazalo se da kocke i kvadri paralelni osama (eng. *axis-aligned box*) pružaju sasvim dovoljno slobode za rad ako se izaberu kao primitivan oblik regiona od kog su sastavljeni svi pokreti. U ostatku rada, regioni će još biti označeni kao *primitive*, kada nije bitan konkretan geometrijski oblik, odnosno biće upotrebljen naziv konkretne geometrijske figure, kada je to od interesa. Konkretan način detekcije je izložen u narednom primeru, a detalji sistema klasa opisani su u odeljku 4.3.

Na slici 4 je prikazan sistem detekcije pokreta "prevlačenja rukom udesno" (eng. *swipe right*). Za prepoznavanje gesture koriste se tri regiona, a kao aktivator je izabrana leva šaka. Redosled regiona kroz koje korisnik treba da provuče levu šaku je prikazan na slici. S obzirom na to da se pokreti delova dela dobijaju 30 puta u sekundi u proseku, ako bi korisnik prebrzo izveo pokret, sistem ne bi zabeležio sve međupozicije desne šake i ne bi aktivirao pokret. Iako bi bilo moguće aproksimirati međupozicije šake interpolacijom, i tako približno utvrditi kuda se ona kretala u proteklom periodu, a samim tim utvrditi da li je ona prošla kroz neke regione, ta mogućnost se ne koristi u aplikaciji KinectCity. Naime, time se izbegavaju situacije u kojima korisnik brzim pokretima neželjeno izazove detekciju pokreta. Ovakve situacije nastaju kao posledica toga što se korisnici uglavnom nisu navikli da koriste ovakav sistem interakcije sa računarom, pa neopaženo ili refleksno izvedu neki inače uobičajeni pokret telom. Zahvaljujući ovakvom pristupu, dovoljno je da sistem čuva relativno mali broj prethodnih pozicija aktivatora tokom vremena, ali se od korisnika zauzvrat očekuju relativno precizni i po pitanju brzine umereni pokreti. Zahvaljujući maksimalnom očekivanom vremenu za izvođenje pokreta, efikasno se izbegava problem kasne ili zaostale detekcije pokreta. Teoretski posmatrano, za dovoljno velike regione postoji mogućnost da se veoma brza kretanja praćenih tačaka prepoznaju kao pokret. U tom slučaju, bilo bi neophodno uvesti i minimalno očekivano vreme izvršenja pokreta. Međutim, prilikom testiranja, nijednom nije zabeležen slučaj lažne detekcije kod brzih pokreta, pa se stoga minimalno očekivano vreme ne koristi prilikom odlučivanja.

U aplikaciji je u nekoliko režima rada potrebno izvesti geste čiji je smisao "sledeći", odnosno "prethodni" (na primer prilikom izbora scene ili građevine). Odlučeno je da se ovi gestovi realizuju pokretima desne šake ulevo, odnosno leve šake udesno, respektivno, ispred i približno u visini grudnog koša. Za detekciju ovih pokreta, ispred avatara (odeljak 3.4) postave se dve grupe od po tri kocke. Jedna grupa služi za detekciju pokreta ulevo, a druga za detekciju pokreta udesno. U toku razvoja, uočena je tendencija korisnika da nesvesno menja visinu na kojoj se nalazi šaka u toku trajanja pokreta, tako što pri kraju pokreta šaku povuče na gore. Zbog toga se često dešavalo da se gest ne detektuje, jer šaka ne prođe kroz poslednju u nizu kocaka. Ovaj problem je rešen tako što veličina kocaka progresivno raste u smeru detekcije pokreta. Time je obezbeđena ispravna detekcija pokreta čak i u situacijama kada korisnik ne obraća dovoljno pažnje na preciznost svojih pokreta.

Za neke od režima potrebno je izvesti gesturu čiji je smisao "nazad", odnosno povratak u prethodni režim. Ova gestura je analogna pritiskanju tastera ESC za poništenje otvorenog dijaloga, bez izmena stanja u aplikaciji. Ova gestura izvodi se spajanjem leve i desne šake (na primer - pljesak dlanovima). Za detekciju ove gesture, za desnu šaku avatara vezana je kocka (slika 4). Ulazak leve šake u tu kocku aktivira prepoznavanje ove gesture.

Za promenu režima kretanja korisnika u sceni u aplikaciji KinectCity, pored komandi na tastaturi moguće koristiti i glasovne komande koje se sastoje od pojedinačnih reči na engleskom jeziku. Zbog velike osetljivosti upotrebljenog sistema za prepoznavanje na šum, učestalost grešaka (lažnih prepoznava-

nja, odnosno neprepoznavanja izgovorenih komandi) raste sa povećanjem rečnika. U cilju smanjenja učestalosti grešaka, bilo je potrebno odabrati što manji broj reči (glasovnih komandi) tako da se po izgovoru (zvučnosti) što više razlikuju. Komande koje je moguće zadati određene su tekućim režimom rada aplikacije. Spisak dozvoljenih glasovnih komandi za svaki režim rada ispisan je u gornjem desnom uglu prozora. Prepoznavanje se vrši tako što se zvuk koji dolazi sa mikrofona analizira u kontinuitetu. Kad sistem za prepoznavanje pozitivno detektuje neku od mogućih reči, aplikaciji se izdaje naredba na isti način kao kad se napravi gestura. Pored toga, kao povratna informacija korisniku, naziv prepoznate komande prikazuje se pri dnu prozora i nakon određenog vremena ukloni.

U svakom režimu rada aplikacije moguće je omogućiti i onemogućiti glasovne komande. Za to se koristi posebna glasovna komanda «voice». Ona je uvek omogućena. Naravno, moguća je pogrešna detekcija i same komande «voice», koja značajno otežava izdavanje glasovnih komandi, što je primećeno u toku razvoja i upotrebe aplikacije.

3.3 *Automatska adaptacija tačke fokusa u režimu stereoskopskog prikaza*

3.3.1 Stereoskopski prikaz

Stereoskopski prikaz podrazumeva formiranje dve slike u istoj sceni (tzv. *stereo-par*), od kojih je jedna namenjena levom, a druga desnom oku. U odnosu na konvencionalne 2D prikazivače, koji prikazuju samo jednu sliku, stereoskopski prikaz korisniku daje osećaj dubine u 3D prostoru. Proces je baziran na biološkom procesu *stereopsije* koji, na osnovu disparitеta slika koje mozak dobija od očiju, omogućava percepciju dubine i trodimenzionalnih tela, kao i procenu razdaljine posmatranih objekata od posmatrača.

Implementacija stereoskopskog prikaza podrazumeva postojanje dve povezane, razmaknute kamere. Povezanost kamera podrazumeva da njhovo kretanje i pozicioniranje u prostoru nije nezavisno, kao što se ni oči živih bića ne mogu pozicionirati nezavisno (sa izuzetkom nekih životinja, poput kameleona, koji upravo imaju mogućnost u određenoj meri nezavisnog kretanja očiju). Kamere moraju biti razmaknute da bi proizvele željeni disparitet u stereo-paru. Dodatno, orijentacija kamera mora biti koordinisana, tako da istovremeno posmatraju istu tačku u 3D prostoru. Treba napomenuti da je postojanje dve kamere konceptualna stvar. Isti rezultat se može postići jednom kamerom koja najpre napravi sliku za jedno, a zatim za drugo oko. Trenutno najpopularnije tehnologije za stereoskopski prikaz su polarizacija svetla i zamračivanje. Kod tehnike polarizacije svetla, prikazivač polarizuje svetlost koja potiče od slika u stereo-paru na različite načine. Korisnik nosi specijalne naočare sa polarizovanim staklima tako da levo, odnosno desno oko vidi samo sliku namenjenu levom, odnosno desnom oku. Kod tehnike zamračivanja (eng. *shutter*), prikazivač naizmenično prikazuje slike iz stereo-para. Korisnik nosi specijalne naočare koje zamrače staklo ispred onog oka kojem nije namenjena slika koja se emituje u tom trenutku.

3.3.2 Implementacija u aplikaciji KinectCity

U razvoju aplikacije KinectCity se pojavio problem dinamičke promene fokusa (objekta koji posmatrač gleda). Naime, posmatrač može da pogledom fokusira bilo koji objekat u sceni. Povratnu informaciju o tome u kom pravcu posmatrač gleda u datom trenutku može da pruži uređaj za praćenje pogleda (eng. eye tracker). Na osnovu te informacije, aplikacija bi mogla da dinamički odredi koji objekat u sceni treba fokusirati. S obzirom na relativno malu dostupnost takvih uređaja, u ovom radu je razmatrano drugačije rešenje problema dinamičke promene fokusa.

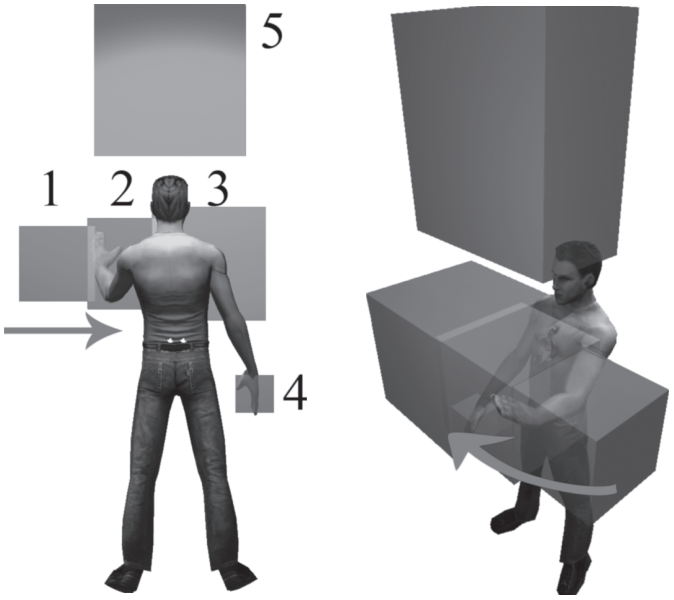
U cilju jednostavnog praktičnog rešenja pomenutog problema, usvojena je pretpostavka da korisnik fokusira najbliži objekat, koji se nalazi u centru prikaza. Za određivanje rastojanja do najbližeg objekta, koristi se tehnika projekcije zraka (eng. ray-casting). Prvi eksperimenti upotrebe ovakvog pristupa u toku razvoja pokazali su da postoje dva problema. Prvi problem je potreba za fokusiranjem i onih objekata koji se nalaze levo, odnosno desno od centra prikaza. Drugi problem je česta i nagla promena tačke fokusa pri kretanju ili promeni pravca gledanja, što izaziva veoma neprijatan osećaj kod korisnika. Kako bi se ovi problemi rešili, uvedene su dve pomoćne tehnike. Najpre, umesto jednog zraka, projektuju se tri zraka, od kojih je jedan u centru prikaza, a preostala dva, periferna, levo i desno od centra prikaza. Periferni zraci polaze iz odgovarajuće kamere u stereo paru i usmereni su prema tački fokusa. Pozicija tačke fokusa se onda dinamički računa kao ponderisana suma pozicija tačaka intersekcije svakog zraka i odgovarajućih objekta u sceni, najbližih posmatraču. Eksperimentalnim putem je utvrđeno da se najbolji rezultati dobijaju kada se centralnom zraku pridruži najveći značaj, što se i moglo očekivati. Drugi problem rešen je interpolacijom prethodne i trenutno izračunate tačke fokusa, čime se izbegavaju nagle, vizualno neprijatne promene tačke fokusa u situacijama kada se neki blizak objekat, u kratkom vremenskom intervalu, nađe blizu centra prikaza.

Iako ne postoji opravdanje (u vidu biološkog procesa), tokom razvoja je uočeno da se dobar vizuelni efekat dobija promenom rastojanja između kamera u zavisnosti od udaljenosti tačke fokusa od posmatrača. Zbog toga je takvo ponašanje implementirano u aplikaciji. Inicijalno se postavi željeno rastojanje između kamera i minimalna (granična) udaljenost posmatrača od tačke fokusa. Ako je udaljenost veća od zadate granične vrednosti, rastojanje između kamera se ne menja. Od trenutka kada udaljenost postane manja, rastojanje postepeno opada, srazmerno udaljenosti posmatrača od tačke fokusa.

3.4 *Avatar (vizuelna povratna informacija od uređaja Kinect)*

Za vreme testiranja funkcionalnosti prepoznavanja gestura (komandi korisnika), primećeno je da postoji izražena potreba za povratnom informacijom o putanji kretanja ruke kojom je korisnik napravio gest i kako izgleda očekivana putanja (u slučaju da korisnik nije napravio pravilan pokret). Ta funkcionalnost realizovana je prikazivanjem avatara koji u potpunosti

preslikava pozicije ključnih delova tela korisnika (onako kako ih uređaj Kinect opaža u prostoru), i prikazivanjem svih primitiva koje odgovaraju trenutno aktivnim (očekivanim) gesturama. Postepenim prolaskom kroz primitive pridružene jednoj gesturi, korisnik dobija vizuelnu povratnu informaciju o očekivanom redosledu tako što se odgovarajuće primitive boje zeleno. Na slici 4 ilustrovan je postupak prepoznavanja gesture kretanjem leve šake s leva udesno.



Slika 4. – *Prikaz avatara, primitiva za detekciju gestura i povratna informacija o pravilnom prolasku kroz primitive. a) Korisnik pomera svoju levu šaku udesno (u smeru strelice), što je uspešno detektovano primitivama (1) i (2). Prolaskom ruke kroz primitivu (3) biće prepoznata odgovarajuća gestura. Primitiva vezana za desnu šaku (4) služi za detekciju gesture poništavanja izbora. Primitiva postavljena iznad i ispred korisnika (5) služi za potvrdu selekcije. b) Pogled na sliku (a) iskoso spreda.*

3.5 *3D kursor*

Položaj kursora u 3D sceni predstavljen je 3D modelom u obliku letećeg tanjira koji prati poziciju desne šake u prostoru. Da bi se posmatraču omogućilo da jednostavnije i preciznije odredi pozicija letećeg tanjira (kursora) u prostoru, vrši se crtanje transparentne siluete tanjira projektovane vertikalno na dole. Preslikavanje pozicije šake (u realnom svetu) u poziciju kursora realizovana je po analogiji sa preslikavanjem iz prozora posmatranja (eng. *window*) u prikazni prozor (eng. *viewport*), koje se standardno koristi u 2D grafici. Preciznije, za preslikavanje se koriste dva kvadra. Jedan kvadar, koji je vezan za korisnikovog avatara (ispred desne ruke, u visini podlaktice, približno obuhvata prostor koji desna ruka može da prebriše bez pomeranja ostatka tela) služi za preslikavanje pozicije šake u normalizovane koordinate, u opsegu [-1, 1], sa koordinatnim početkom u centru kvadra. Drugi kvadar se nalazi iznad tačke preseka centralnog zraka stereo-kamere (odeljak 3.3) i terena, i služi za preslikavanje pozicije šake iz normalizovanih koordinata u koordinate kursora u prostoru scene. Kvadar se skalira srazmerno razdaljini posmatrača od tačke preseka, tako da pokriva značajan deo vidnog polja ispred posmatrača. Na taj način je realizovano potpuno intuitivno pozicioniranje kursora u sceni.

U vreme razvoja aplikacije primećen je problem nestabilnosti podataka koje dostavlja Kinect-a. Naime, često se dešava da Kinect na trenutak saopšti značajno drugačiji položaj ruke u odnosu na stvarni. To se manifestuje čestim trzanjem (poskakivanjem) kursora u sceni, čak i u situacijama u kojima je desna šaka potpuno nepokretna. Taj problem rešen je filtriranjem, odnosno linearnom interpolacijom prethodne i novodobijene pozicije, sa težinskim faktorom 0.98 u korist prethodne pozicije. Vrednost težinskog faktora je utvrđena eksperimentalnim putem. Mana pristupa je osetno smanjenje brzine odziva, što se u praksi pokazalo prihvatljivim spram neprijatnog trzanja kursora.

3.6 *Režimi rada*

Kako bi se olakšala implementacija izvršenja korisnikovih zahteva, interakcija sa korisnikom i dalji razvoj, aplikacija je realizovana tako da različita ponašanja predstavlja većim brojem stanja, kojima upravlja konačni automat (eng. *finite state machine*). Neka od tih stanja su: odabir scene, slobodno kretanje, vožnja motora, odabir i razmeštanje građevinskih objekata po sceni. U zavisnosti od stanja, korisniku je na raspolaganju odgovarajući skup gestura kojima izdaje komande.

3.6.1 Odabir scene

Kada korisnik bira scenu, na raspolaganju su gestovi kojima prelazi na sledeću odnosno na prethodnu scenu, kao i gest za potvrdu da bira trenutno prikazanu scenu. Gestovi kojima korisnik menja scenu opisani su odeljku 3.2. Za potvrdu izbora, iznad i ispred glave avatara nalazi se još jedna kocka kroz koju korisnik treba da provuče ruku odozgo na dole.

Nakon izbora scene, a u cilju bržeg pristupa njenom željenom delu, korisniku se prikaže cela scena, a njegov zadatak je da upotrebom 3D kursora (odeljak 3.5) označi željeno mesto na terenu. Kursorom upravlja desnom šakom. Slično kao i kod izbora scene, kada je zadovoljan pozicijom kursora, korisnik treba da levom šakom napravi gest prolaska kroz kocku koja se nalazi iznad i ispred njegove glave. Ovim odabiranjem aplikacija prelazi u stanje slobodnog kretanja (razgledanja).

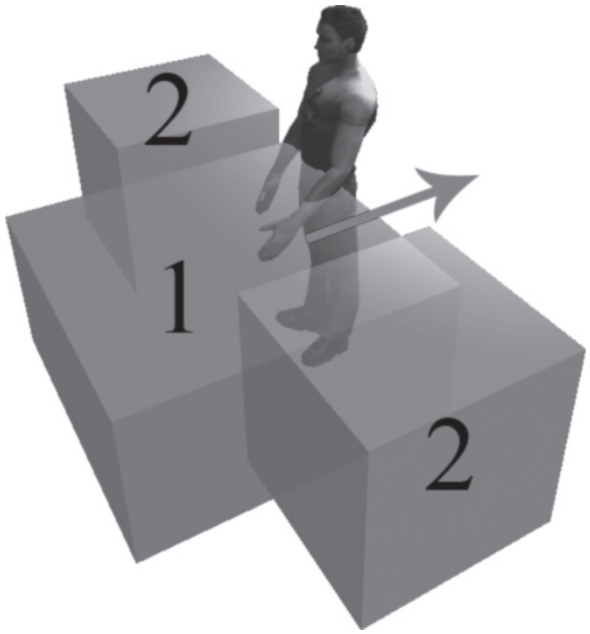
3.6.2 Slobodno kretanje

U stanju slobodnog kretanja, korisnik na raspolaganju ima više različitih načina kretanja kroz prostor. Korisnik pristupa željenom načinu kretanja tako što izgovori naziv datog režima, što predstavlja glasovnu komandu prelaska u dati režim. Tada se ispred avatara pojave one primitive za detekciju gestova koje su predviđene za dati režim kretanja, a ostale nestanu. U nastavku su ukratko opisani načini kretanja. U zagradama je naveden naziv glasovne komande koji aktivira prelazak u dati režim.

Translacija (Pan)

U ovom režimu kretanja, levo od avatara nalazi se kvadar koji služi za translatorno kretanje u horizontalnoj ravni. Kori-

snik vrši translaciju tako što levu šaku spusti u kvadar, s gornje strane. Sva dalja pomeranja šake, dok se nalazi u datom kvadru, interpretiraju se kao translacije u horizontalnoj ravni (slika 5). Ovo je implementirano tako što se prati pomeraj korisničke ruke koja se nalazi u kvadru u odnosu na njenu prethodnu poziciju. Pomeraj se skalira u zavisnosti od elevacije posmatrača (veća elevacija - veći pomeraj), a vertikalna komponenta se ignoriše. Na taj način korisnik sebe može da privuče ka odnosno udalji od željene pozicije na terenu. Pokret donekle podseća da veslanje levom rukom.



Slika 5. – *Translacija u horizontalnoj ravni upotrebom leve ruke. Pozicija leve šake prati se kontinuirano sve dok se nalazi unutar kvadra (1) namenjenog gesturi translacije. Kvadri levo i desno od avatara (2) služe za ulaz u režim vožnje motocikla.*

Letenje (Fly)

U ovom režimu kretanja korisnik na raspolaganju ima dva kvadra koji se nalaze s desne strane avatara (prvi u visini glave, drugi u visini kukova) i služe za promenu elevacije (prvi za povećanje, a drugi za smanjenje). Ulaskom desne šake u neki od ova dva kvadra, korisnik započinje postepenu promenu sopstvene elevacije, a izlaskom šake je zaustavlja. Nagib kamere se automatski podešava, tako da uvek bude usmerena na dole. Kada je posmatrač blizu horizontalne ravni, kamera naginje blago na dole, a pri minimalnoj elevaciji vektor pogleda je paralelan horizontalnoj ravni. Sa povećanjem visine posmatrača progresivno se povećava nagib kamere dok vektor pogleda ne postane normalan na horizontalnu ravan.

Rotacija (Rotate)

Korisnik može da izvede dve vrste rotacije: (1) oko vertikalne ose vezane za posmatrača i (2) oko vertikalne ose koja prolazi kroz tačku preseka vektora pogleda u centru prikaza (odeljak 3.3.2) i horizontalne ravni ($y = 0$). Za kontrolu rotacije na raspolaganju je kvadar pozicioniran ispred avatara. U zavisnosti od toga koju ruku koristi, korisnik će vršiti drugačiji vid rotacije (leva ruka za (1), desna za (2)). Kao u slučaju translacije, prati se pomeraj korisnikove ruke kroz kvadar, pri čemu se prati samo pomeraj po X osi, dok se zanemaruju ostale dve komponente ($y = 0$, $z = 0$) .

Zumiranje (Zoom)

Korisniku je omogućena i pravolinijska translacija u pravcu vektora pogleda (u centru prikaza). Ovaj način kretanja izvodi se upotrebom dve ruke. Ispred avatara, u visini grudnog koša, nalazi se kvadar dovoljno širok da obuhvati obe šake, kada su ruke razmaknute. Korisnik kretanje vrši tako što obe šake smesti u kvadar. Daljim povećanjem ili smanjenjem rastojanja između šaka korisnik vrši translaciju ka ili od terena. Promena rastojanja se zatim koristi kao pomeraj. Ovaj način kretanja oponaša gesture "smanjiti" (eng. pinch) i "uvećati" (eng. zoom) na ekranima osetljivim na dodir.

3.6.3 Izgradnja objekata

Korisnik bira objekat koji želi da izgradi iz jedne od paleta objekata za gradnju. Objekti su raspoređeni po paletama na osnovu svoje kategorije, tako da svaka kategorija poseduje sopstvenu paletu. Postoje tri kategorije objekata za gradnju: zgrade, dekoracija (fontane i spomenici), i infrastruktura (klupe, hidranti, telefonske govornice itd). Postoji posebna paleta za odabir kategorije.

Kada želi da sagradi (odnosno postavi) određeni objekat, korisnik pristupa paleti za odabir kategorija tako što desnom šakom prođe odozgo na dole kroz tri vertikalno poređane kocke, koje se nalaze ispred avatara, kao da sa vrha prozora spušta meni. Paleta kategorija se tada otvara i aplikacija prelazi u stanje odabira kategorije predmeta za gradnju. Primer izgleda palete kategorija prikazana je na slici 3a, gde je u prednjem planu kategorija infrastrukturnih objekata, predstavljena kabinom telefonske govornice.

Navigacija kroz palete se vrši na isti način kao i odabir scene: pokretima ruku se prelazi na sledeći odnosno prethodni izabrani predmet, što je vizuelno prikazano rotacijom palete. Zatim se izdaje potvrda prolaskom ruke kroz kocku koja se nalazi iznad glave (odeljak 3.2).

Nakon što korisnik odabere objekat za izgradnju i potvrdi izbor, paleta se zatvara i aplikacija prelazi u režim raspoređivanja izabranog objekta na teren. U tom režimu, objekat prati poziciju 3D kursora sve dok ne dodirne tlo. Tada se smatra da je objekat raspoređen i prekida praćenje kursora. Indikacija da je objekat u koliziji sa drugim objektima pruža se promenom boje kursora u crveno, i tada nije moguće raspoređivanje objekta. Raspoređen objekat može da se premesti tako što se kursor kratko zadrži neposredno iznad objekta. Tada objekat počne da prati poziciju kursora i dalje premeštanje se vrši isto kao i raspoređivanje novog objekta.

3.6.4 Vožnja motocikla

Pored tehnički-orijentisanih načina razgledanja scene, pojavila se potreba za kreativnim i prirodnim načinom da kori-

snik, iz ljudske perspektive, osmotri grad koji stvara. Kao odgovor na to javila se ideja da se omogući vožnja motocikla. U režimu slobodnog kretanja, sa leve i desne strane avatara postoje dve kocke koje se prostiru od tla do kukova. Kako bi započeo vožnju, korisnik proizvoljnom nogom treba da napravi pokret kojim bi fiktivno opkoračio motocikl, kao da seda na njega. Prilikom tog pokreta, korisnik stopalom ulazi u jednu od kocki, kamera se spušta do visine koja odgovara poziciji glave vozača na motociklu, a aplikacija prelazi u stanje vožnje motocikla (slika 3c).

Za samo upravljanje motociklom se ne koriste nikakve primitive, već se kompletna korisnička interakcija vrši na osnovu međusobnog nagiba i rastojanja pojedinih čvorova u detektovanom položaju skeleta korisnika. Korisnik stavljanjem ruku ispred sebe, poput držanja ručki motocikla, dodaje gas (tj. povećava moment zadnjem točku). Srednja vrednost rastojanja (po Z osi, X i Y se zanemaruju) korisnikovih šaka od njegovog grudnog koša se koristi kao intenzitet momenta koji se dodaje točku. Kako se motor ne bi kretao dok korisnik drži ruke pored tela (neutralni položaj), ovo rastojanje se dodaje kao moment točka samo ukoliko je veće od određene, empirijski utvrđene vrednosti. Kočenje korisnik vrši povlačenjem laktova unazad (iza zamišljene vertikalne ravni koju čini torzo), prilikom čega se računa srednja vrednost rastojanja (duž Z ose) laktova od grudnog koša. Skretanje korisnik postiže savijanjem tela u kukovima na levo ili desno. Ugao skretanja se određuje kao ugao između vektora koji se proteže od glave korisnika pa do sredine kukova u odnosu na vertikalnu (Y) osu.

U toku testiranja aplikacije, povremeno se javljao problem detekcije kolizije točkova motocikla sa terenom i dešavalo se da motocikl ostane zaglavljen u terenu (točak propadne kroz teren), naročito u situaciji kada koliziji prethodi pad motocikla sa veće visine. Rešenje ovog problema je u potpunosti prepušteno korisniku. Korisnik na raspolaganju ima dve kocke iznad avatara, s leve i desne strane, kojima može da menja poziciju motocikla na terenu. Korisnik bi prolaskom ruke kroz levu kocku povećao vertikalnu komponentu pozicije motora za neku relativno malu vrednost i na taj način se "odglavio". Prolaskom ruke kroz desnu kocku korisnik bi postavio motocikl na centar terena u horizontalnoj ravni i blago iznad terena. Izlazak iz režima vožnje motocikla vrši se na isti način kao i ulazak.

4. DETALJI IMPLEMENTACIJE

U ovom poglavlju izloženi su neki detalji implementacije aplikacije KinectCity. Objašnjen je način formiranja stereoskopskog prikaza za projekciju na platno, način primanja glasovnih komandi, način detekcije pokreta i tehnički detalji aplikacije.

4.1 *Prikazivanje aplikacije u stereoskopskom režimu*

Da bi se formirao stereoskopski prikaz na paru projektora, aplikacija otvara prozor veličine 3840 x 1080 piksela, bez dekoracija, i rasporedi ga tako da se jedna njegova polovina preslikava na levi, a druga na desni projektor. Scena se crta dva puta, za svaku kameru u stereo-paru po jednom. Pri svakom crtanju, odgovarajući prikazni prozor se pozicionira na levu odnosno desnu polovinu prozora aplikacije, tako da se slika leve, odnosno desne kamere preslikava na projektor čija je slika namenjena levom, odnosno desnom oku, respektivno. Za potrebe kreiranja prozora aplikacije korišćene su funkcije *SetWindowPos* i *FindWindow* iz sistemske biblioteke *user32.dll* u kojoj se nalaze funkcije za upravljanje grafičkim korisničkim interfejsom.

Za implementaciju stereoskopskog prikaza (odeljak 3.3.2) u Unity3D okruženju kreirana je C# klasa koja predstavlja stereo-kameru i koja se parametrizuje sledećim podacima:

- Objekti leve i desne kamere (Unity objekti sa komponentom *Camera*) sa istim parametrima prikaza
- Faktor interpolacije između prethodne i trenutno izračunate tačke fokusa (brzina promene fokusa)
- Minimalna udaljenost tačke fokusa od stereo-kamere
- Ponderi uticaja zraka projektovanih u cilju određivanja tačke fokusa
- Željena udaljenost očiju
- Granična udaljenost tačke fokusa od stereo-kamere

4.2 *Detekcija zvuka*

Za potrebe detekcije zvuka u aplikaciji KinectCity, razvijena je posebna C# aplikacija za prepoznavanje glasovnih komandi i Unity C# skripta koja preuzima rezultate te aplikacije. Korišćena verzija Unity 3D razvojnog okruženja koristi Microsoft .NET platformu, verzija 3.5, a biblioteke za prepoznavanje glasovnih komandi zahtevaju verziju 4.0. Zbog toga je prepoznavanje glasovnih komandi realizovano u posebnoj aplikaciji. Prilikom inicijalizacije (metoda Awake), skripta napravi lolakni tok podataka, pokrene aplikaciju kao zaseban proces, preusmeri njen standardni izlaz u napravljeni lokalni tok podataka i na taj način dohvata podatke koji pristižu kao tekstualni zapis. Aplikacija formira rečnik reči koje je moguće prepoznati, inicijalizuje sistem za prepoznavanje govora i mikrofon. Rečnik je realizovan kao tabela parova reči i pragova (donja granica) pouzdanog prepoznavanja. Pragovi pouzdanog prepoznavanja reči su potrebni u cilju izbegavanja lažnog prepoznavanja komandi uzrokovanih šumom ili lošim izgovorom. Upotreba pragova prepoznavanja će biti objašnjena u nastavku.

Unity skriptu čini klasa SoundInput i njoj pridruženo nabrajanje SoundCommand. Putem klase SoundInput programer može da ispita da li je izgovorena glasovna komanda zadata konkretnom vrednošću tipa SoundCommand. Programer ima odgovornost da izvrši ažuriranje stanja svaki put kad se promeni režim rada aplikacije, odnosno kada se promene moguće glasovne komande. Nabrajanje SoundCommand sadrži po jedan identifikator za svaku podržanu glasovnu komandu.

Aplikacija za detekciju zvuka i prepoznavanje govora sastoji se od klase MicrophoneSoundInputApp. Klasa ima minimalistički interfejs prema korisniku, koji se sastoji od dve metode: za inicijalizaciju i za deinicijalizaciju. Inicijalizacija obuhvata formiranje rečnika, inicijalizaciju sistema za prepoznavanje govora iz Microsoft .NET Framework 4.0 biblioteke *System.Speech*, kao i povezivanje mikrofona sa aplikacijom.

Klase iz biblioteke *System.Speech* koje se koriste su: *SpeechRecognitionEngine*, klasa koja omogućava rad sa instaliranim servisima za prepoznavanje govora na Windows operativnom sistemu; *RecognizerInfo*, koja pruža informacije o servisu za prepoznavanje govora; *Choices*, koja sadrži kolekciju izraza koji predstavljaju osnovni element gramatike jezika; *GrammarBuilder*, koja služi za jednostavnu inicijalizaciju objekata klase Grammar; i *Grammar*, koja pruža podršku za upravljanje podacima o gramatici u vreme izvršenja.

Inicijalizacija sistema za prepoznavanje sprovodi se na sledeći način. Najpre se dohvati jedan od dostupnih RecognizerInfo objekata, pomoću SpeechRecognitionEngine klase. Zatim se inicijalizuje gramatika tako što se najpre sve reči i eventualno rečenice čije prepoznavanje je potrebno omogućiti dodaju u objekat klase Choices. Nakon toga napravi se GrammarBuilder objekat na osnovu kulture, grada i regije koji su dostupni u objektu tipa RecognizerInfo i u njega se doda odgovarajući objekat klase Choices. Pomoću objekta GrammarBuilder napravi se objekat Grammar koji sadrži sve podatke potrebne za prepoznavanje zadatih reči odgovarajućeg jezika.

Glavna nit aplikacije za prepoznavanje glasovnih komandi je uspavana sve vreme, osim kad se vrši inicijalizacija i deinicijalizacija. Prepoznavanje se vrši u posebnoj niti. Prilikom prepoznavanja govora, uz svaki prepoznati element, biblioteka System.Speech dostavlja i stepen pouzdanosti. Kada sistem za prepoznavanje prepozna, odbaci ili pretpostavi da je prepoznao neku reč, poziva se odgovarajući od tri rukovaoca događajima. Rukovaoci za odbačenu reč ili reč za koju se pretpostavlja da je prepoznata imaju prazno telo. Samo rukovalac za prepoznatu reč vrši obradu. Obrada se svodi na ispitivanje pouzdanosti prepoznavanja koju dostavlja biblioteka System.Speech. Ako je pouzdanost prepoznavanja ispod minimalne dozvoljene za datu reč (dohvata se iz rečnika), rezultat se odbija. U suprotnom rezultat se prihvata i na standardni izlaz šalje tekst u formatu "Recognized: <ime reči>". Time se efektivno šalje poruka SoundInput skripti (u aplikaciji KinectCity) da je nova reč prepoznata. Rukovalac događajima u SoundInput skripti pročita navedeni tekst i lokalno ga zapamti. Programer ima odgovornost da u odgovarajućem trenutku ispita koja reč je izgovorena.

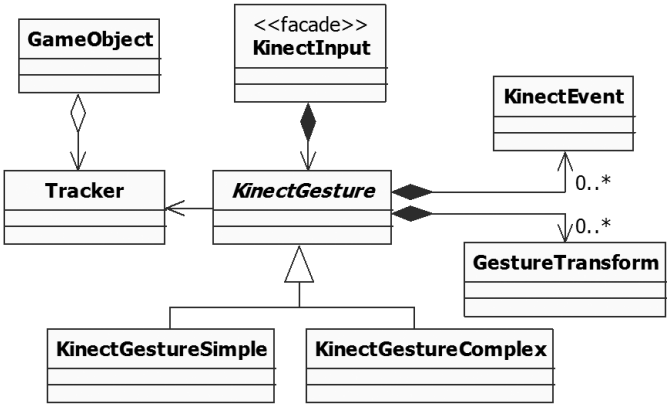
4.3 *Implementacija detekcije pokreta*

Za povezivanje Kinect SDK i Unity, korišćena je biblioteka *Kinect Wrapper Package for Unity3D* [14]. Ova biblioteka omogućava manipulaciju Kinect uređajem iz samog Unity projekta i, preko aktivacije Kinect SDK koji se implicitno dešava u kodu prilikom inicijalizacije, dostavlja pozicije delova tela u prostoru. Iako ova biblioteka daje programeru mogućnost da veoma jednostavno sazna pozicije karakterističnih tačaka tela u prostoru, ipak ne daje nikakve informacije o tome kakvi su pokreti izvršeni, niti ima mogućnost da interpretira njihov smisao. Zbog toga je funkcionalnost detekcije pokreta realizovana kroz sledeće tri klase, koje su posebno razvijene za aplikaciju KinectCity: KinectInput, GestureTransform i Tracker. Kao što je napomenuto u poglavlju 2, izabrana tehnika za detekciju pokreta je segmentacija. Uloga svake od klasa je opisana u nastavku, a uprošćen dijagram klasa dat je na slici 6.

- **KinectInput**: glavna klasa podsistema za detekciju pokreta. Implementirana je upotrebom projektnog uzorka *fasada*. Ova klasa omogućava programeru da kreira i uništava pokrete za vreme trajanja aplikacije. Može da stvara jednostavne i kompleksne pokrete, kao i da ispituje trenutno stanje svakog stvorenog pokreta (da li je izvršen). Sve akcije vezane za pokrete obavljaju se posredstvom ove klase.
- **GestureTransform**: klasa zadužena za zadavanje 3D transformacija koje se primenjuju na primitive koje čine pokret. Sadrži informacije poput veličine i relativne udaljenosti primitive od svog roditeljskog (referentnog) objekta. Pošto se koriste kvadri paralelni osama, nije bilo potrebe čuvati informacije o njihovoj rotaciji.
- **Tracker**: klasa koja obezbeđuje da neki Unity GameObject ima mogućnost da bude posmatran kao aktivator nekog od pokreta. Klasa interno prati i pamti poziciju tog objekta tokom vremena. Zapamćene pozicije kasnije se koriste prilikom ispitivanja da li je određen pokret izvršen.

Pomoćne klase, sakrivene od programera su:

- **KinectGesture**: bazna klasa za opis pokreta.
- **KinectGestureSimple**: klasa zadužena za opis pokreta sastavljenih od samo jedne primitive.
- **KinectGestureComplex**: klasa zadužena za opis pokreta sastavljenih od više primitiva.
- **KinectEvent**: struktura koja čuva informacije o trenutnom stanju gesture i koju koristi klasa KinectGesture.



Slika 6. – *Dijagram klasa za definisanje i detekciju pokreta.*

U listingu 1 je prikazan deo koda koji vrši pravljenje i registrovanje gesture za detekciju rotacije ulevo. Desna šaka koristi se kao aktivator. Desno rame koristi se kao roditeljski (referentni) objekat za grupu primitiva koje definišu gesturu. Pozicije i veličina primitiva su definisane transformacijama, tako da budu raspoređene u prostoru kao što je prikazano na slici 4. Definisana gestura se potom prijavljuje klasi KinectInput, koja stvara odgovarajući objekat tipa KinectGesture (konkretno - KinectGestureComplex), sa zadatim vremenskim ograničenjem detektovanja pokreta.

Listing 1: Registrovanje gesture za detekciju rotacije ulevo.

```
GameObject desna_saka = GameObject.Find("23_Hand_Right");
GameObject desno_rame = GameObject.Find("20_Shoulder_Right");
GestureTransform gtr_zdesna = new GestureTransform(new Vector3(0, 0, 4), Vector3.one * 3);
GestureTransform[] gtr_zdesna_delovi = {
        new GestureTransform(new Vector3(1f,0,0), Vector3.one),
        new GestureTransform(new Vector3(0,0,0), Vector3.one*1.2f),
        new GestureTransform(new Vector3(-1,0,0), Vector3.one*1.5f)
};
KinectInput.AddGesture("Rotate Left", desno_rame, desna_saka, gtr_zdesna, gtr_zdesna_delovi, max_trajanje);
```

Listing 2: Primer načina utvrđivanja ispunjenja očekivanih korisnikovih akcija

```
if (Input.GetKeyDown(KeyCode.Q) || KinectInput.IsDone("Rotate Left")) { ... }
if (KinectInput.IsInside("kocka")) { ... }
```

U zavisnosti od trenutno aktivnog režima rada aplikacije, uključuju se potrebne i isključuju nepotrebne gesture.

Da bi se održala doslednost sa postojećim Unity klasama slične namene, implementacija klase KinectInput izvedena je po uzoru na Unity klasu - Input. Zbog toga mehanizam reagovanja na gesture nije baziran na događajima, već se zasniva na ispitivanju stanja kada je to od interesa. Nakon što se aktivira pokret, stanje očekivane gesture se može proveriti preko KinectInput klase. U listingu 2 prikazan je primer koji ilustruje postupak utvrđivanja korisnikovih akcija u dve različite situacije.

4.4 *Tehnički detalji*

Aplikacija KinectCity razvijena je upotrebom besplatne verzije Unity 3D 4.1 okruženja na Windows 7 operativnom sistemu. Skripte za kontrolu aplikacije su pisane u jeziku C# upotrebom razvojnog okruženja Visual Studio 2013. 3D model koji je korišćen za prikazivanje korisnikovog avatara kao i modeli motocikla i letećeg tanjira (3D kursora), preuzeti su sa TurboSquid portala pod *royalty free* licencom [15, 16, 17].

5. ZAKLJUČAK

U radu je predstavljena aplikacija KinectCity, u kojoj korisnik može da interaktivno gradi i slobodnim kretanjem razgleda grad. Korisnik interaguje pokretima ruku i nagibom trupa, a detekcija pokreta se vrši upotrebom uređaja Kinect. Korisnik ima mogućnost i zadavanja glasovnih komandi kojima se kontrolišu režimi rada, odnosno režimi prepoznavanja pokreta. S obzirom na to da je detekcija pokreta podložna greškama, u cilju pružanja jasne povratne informacije korisniku o tome koji pokret telom je prepoznat, aplikacija prikazuje animiranu figuru ljudskog tela. Animacija figure vrši se saglasno prepoznatim pokretima. Slično tome, za prepoznate glasovne komande ispiše se njihov naziv. U aplikaciji postoje realizovane dve scene (dnevna i noćna), a dodavanje novih scena je veoma jednostavno. Aplikacija je osposobljena za stereoskopski prikaz scene, zahvaljujući čemu se kod korisnika značajno pojačava utisak stvarne manipulacije objektima u 3D prostoru. Funkcionalnost i efektnost aplikacije je ličnim učešćem potvrdilo više desetina studenata Elektrotehničkog fakulteta u Beogradu.

Dalji razvoj aplikacije bi mogao da se odvija u sledećim pravcima. Najpre, tokom razvoja je uočeno da trenutni način izdavanja komandi pokretima ruku i delovima tela nije u željenoj meri jednostavan i intuitivan za sve korisnike. Zbog toga bi bilo korisno uvođenje profila korisnika, a svaki korisnik bi mogao da, kroz jednostavan proces treninga, podešava način na koji će aplikacija prepoznavati karakteristične pokrete kojima se izdaju komande. Osim toga, od interesa bi bilo da se prouči i ustanovi da li postoje karakteristični pokreti za izdavanje određenih komandi koje bi većina korisnika prihvatila kao intuitivne. Na osnovu tih rezultata mogla bi da se razvije posebna biblioteka pokreta sa ciljem da postane standard u interakciji između čoveka i računara. Uočen problem pogrešnog prepoznavanja glasovnih komandi bi bilo moguće delimično rešiti uvođenjem složenih naziva komandi, sastavljenih od više reči, da bi se smanjila verovatnoća pogrešnog prepoznavanja.

ZAHVALNICA

Autori se zahvaljuju kompaniji Nordeus na doniranoj opremi koja je korišćena u razvoju aplikacije KinectCity. Autori se posebno zahvaljuju Jovanu Stojšiću na pozajmljenom uređaju Kinect XBox 360 koji je pomogao da se razvoj i testiranje aplikacije značajno ubrza. Autori se takođe zahvaljuju gospodinu MacCormick na dozvoli za korišćenje slika iz njegovih publikacija. Učešće Đ. Đurđevića u ovom radu je delimično finansirano sa projekata TR32039 i TR32047 Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije.

LITERATURA

[1] [online] https://www.google.com/glass/start/ , poslednji pristup 28.11.2014.
[2] [online] http://www.sony.com/SCA/company-news/press-releases/sony-computer-entertainment-america-inc/2014/sony-computer-entertainment-announces-project-morp.shtml, poslednji pristup 28.11.2014.
[3] [online] http://www.oculus.com/rift, poslednji pristup 28.11.2014.
[4] [online] http://en.wikipedia.org/wiki/Cave_automatic_virtual_environment, poslednji pristup 28.11.2014.
[5] [online] https://www.leapmotion.com/product, poslednji pristup 28.11.2014.

[6] [online] https://www.microsoft.com/en-us/kinectforwindows, poslednji pristup 28.11.2014.
[7] [online] http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html, poslednji pristup 28.11.2014.
[8] Zhengyou Z., "Microsoft Kinect Sensor and Its Effect", *IEEE MultiMedia*, vol.19, no. 2, pp. 4-10, April-June 2012, doi:10.1109/MMUL.2012.24
[9] Izadi S., Kim D., Hilliges O., Molyneaux D., Newcombe R., Kohli P., Shotton J., Hodges S., Freeman D., Davison A., Fitzgibbon A., "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera", Proceedings of the 24th annual ACM symposium on User interface software and technology, pp. 559-568, 2011, doi:10.1145/2047196.2047270
[10] John MacCormick, "How does the Kinect work?", Dickinson College Math/CS Chat, Sep. 2011, http://users.dickinson.edu/~jmac/selected-talks/kinect.pdf, poslednji pristup 03.03.2015.
[11] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, "Real-Time Human Pose Recognition in Parts from Single Depth Images", CVPR '11 Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1297-1304, 2011.
[12] L. Breiman, "Random Forests", Machine Learning, vol. 45, no. 1, pp. 5-32, 2001, doi: 10.1023/A:1010933404324
[13] [online] N. GIllian, "Dynamic Time Wrapping", http://www.nickgillian.com/wiki/pmwiki.php/GRT/DTW, poslednji pristup 28.03.2015.
[14] [online] http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK, poslednji pristup 20.03.2015.
[15] [online] FREE Vincent 3D model (rigged), TurboSquid, http://www.turbosquid.com/FullPreview/Index.cfm/ID/515398, poslednji pristup 20.03.2015.
[16] [online] Motorcycle model, TurboSquid, http://www.turbosquid.com/FullPreview/Index.cfm/ID/701893, poslednji pristup 20.03.2015.
[17] [online] uuffoo model, TurboSquid, http://www.turbosquid.com/FullPreview/Index.cfm/ID/167887, poslednji pristup 20.03.2015.

Dario Mirović, Univerzitet u Beogradu - Elektrotehnički fakultet, student osnovnih studija
Kontakt: dariomirovic@gmail.com
Oblasti interesovanja: računarska grafika, veštačka inteligencija u računarskim igrama, virtuelna realnost, framework & engine development

Dragan Okanović, Univerzitet u Beogradu - Elektrotehnički fakultet, student osnovnih studija
Kontakt: dragan.okan@gmail.com
Oblasti interesovanja: računarska grafika, fotorealistično renderovanje, implementacija efekata, GPGPU, virtuelna realnost

Jovan Radivojša, Univerzitet u Beogradu - Elektrotehnički fakultet, student osnovnih studija
Kontakt: jovan.radivojsa@gmail.com
Oblasti interesovanja: Računarska grafika, razvoj video igara, framework & engine development, virtuelna realnost

Nemanja Lučić, Univerzitet u Beogradu - Elektrotehnički fakultet, student osnovnih studija
Kontakt: lucic992@gmail.com
Oblasti interesovanja: računarska grafika, proceduralno generisanje sadržaja, paralelno programiranje, razvoj video igara

Dr Đorđe Đurđević, Univerzitet u Beogradu - Elektrotehnički fakultet
Kontakt: djordje.djurdjevic@etf.bg.ac.rs
Oblasti interesovanja: Računarska grafika, virtuelna realnost, interakcija čovek-računar, objektno orijentisana analiza i dizajn, digitalna obrada slike,e-obrazovanje.