SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 68

INTERAKTIVNO UPRAVLJANJE SADRŽAJEM VIRTUALNE SCENE: NADZORNO UPRAVLJAČKI SUSTAV

Alen Kralj

Zagreb, lipanj 2010.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 68

INTERAKTIVNO UPRAVLJANJE SADRŽAJEM VIRTUALNE SCENE: NADZORNO UPRAVLJAČKI SUSTAV

Alen Kralj

Zagreb, lipanj 2010.

Na ovu stranicu stavite izvornik zadatka završnog rada.

Zahvaljujem se mentorici prof. dr. sc. Željki Mihajlović na izuzetnoj dostupnosti, korisnim savjetima i prilici da radim s izuzetnim ljudima na zanimljivim i poučnim temama.

Također zahvaljujem se mr. sc. Siniši Popoviću na brojnim savjetima prilikom razvoja ovog sustava.

Sadržaj

Uvod ... 1
1. Virtualna stvarnost ... 2
1.1. Uređaji za virtualnu stvarnost ... 3
1.2. Računalna simulacija ... 6
1.3. Primjene ... 6
2. Terapija izlaganjem ... 8
2.1. Terapija izlaganjem pomoću virtualne stvarnosti ... 8
3. Sustav za nadzor i upravljanje virtualnom scenom ... 10
3.1. Koncept rada sustava ... 13
3.2. Komunikacija ... 16
3.3. Biblioteka poruka ... 19
3.4. Model terapeut aplikacije ... 25
3.5. Korisničko sučelje terapeut aplikacije ... 30
Zaključak ... 37
Literatura ... 38

Uvod

U ovom radu biti će opisan nadzorno upravljački sustav koji je oblikovan za potrebe terapije izlaganjem pomoću virtualne stvarnosti. Kako bi se shvatili osnovni koncepti virtualne stvarnosti u poglavlju 1 spomenuto je kako se ostvaruje virtualna stvarnost, te koji uređaji su potrebni za njeno ostvarenje. Poglavlje 2 opisuje terapiju izlaganjem i zašto su tehnologije virtualne stvarnosti zanimljivo područje za ostvarenje takve vrste terapije. Konačno u poglavlju 3 detaljno je opisan nadzorno upravljački sustav. Naveden je osnovni koncept rada cijelog sustava za terapiju izlaganjem pomoću virtualne stvarnosti, komunikacija i poruke koje se koriste u sustavu, te model i korisničko sučelje nadzorno upravljačkog sustava.

Uz rad priložen je medij s implementacijom nadzorno upravljačkog sustava.

1. Virtualna stvarnost

Virtualna stvarnost je skup tehnologija kojima se korisnikova slika stvarnosti nastoji što potpunije zamijeniti slikom virtualnog okruženja [1]. Rezultat simulacije virtualne stvarnosti dobivene uz pomoć računalnog sklopovlja zamjenjuje sliku stvarnosti. U idealnom slučaju korisnikova čula osjećaju samo virtualne podražaje proizvedene računalom, a uz to ostvaren je izravan unos korisnikovih pokreta u računalo.

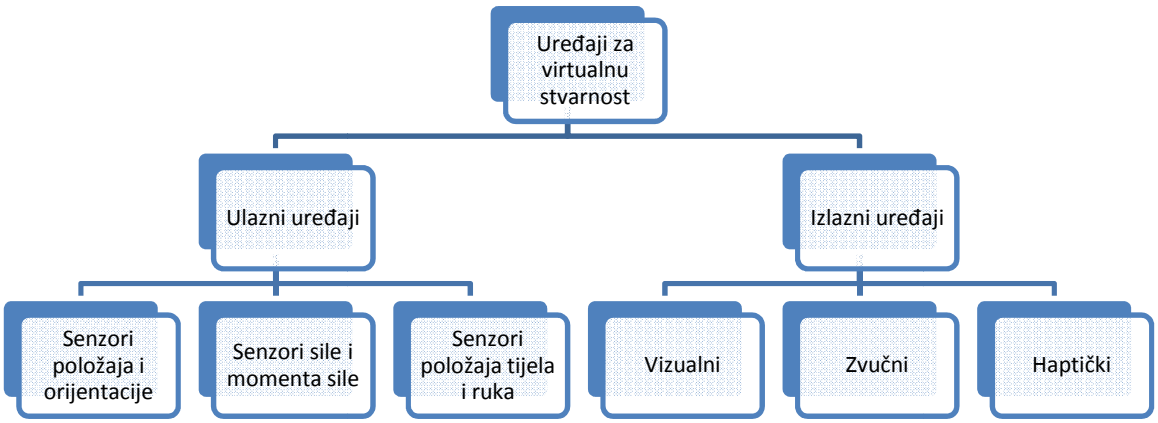
Virtualna stvarnost uključuje najrazličitije ulazno/izlazne uređaje koji korisnika izravno povezuju s računalom i omogućuju neposrednu interakciju korisnika i računala. Većina današnjih okruženja virtualne stvarnosti su primarno fokusirana na vizualan doživljaj virtualnog okruženja koji ostvaruju putem zaslona računala ili posebnih stereoskopskih zaslona. Često uz vizualnu prezentaciju dodaje se i zvučna kroz zvučnike ili slušalice. Međutim samo naprednije simulacije virtualne stvarnosti koriste dodatne prezentacijske sustave, kao što su primjerice haptički sustavi koji omogućavaju simulaciju dodira, povratne sile ili pomične platforme [2]. Slika 1 prikazuje osnovnu petlju principa rada virtualne stvarnosti.

Slika 1 Osnovna petlja principa rada virtualne stvarnosti.

Korisnik se nalazi u zatvorenoj petlji povezan s računalom pomoću ulaznih i izlaznih jedinica. Pomoću ulaznih jedinica korisnik daje informacije položaja i orijentacije unutar 3D prostora na temelju kojih se provodi računalna simulacija, te rezultat računalne simulacije se prikazuje korisniku pomoću izlaznih uređaja.

1.1. Uređaji za virtualnu stvarnost

Uređaje za virtualnu stvarnost dijelimo prvo na ulazne i izlazne, te dalje na vrste s obzirom na senzore kod ulaznih, odnosno prezentaciju kod izlaznih uređaja. Slika 2 prikazuje osnovnu klasifikaciju uređaja koji se koriste za realizaciju virtualne stvarnosti.



Slika 2 Klasifikacija uređaja za virtualnu stvarnost.

U nastavku slijede detaljniji opisi vrsta ulaznih i izlaznih uređaja.

Senzori položaja i orijentacije se često nazivaju slijednici. Oni mogu raditi na različitim principima, kao što su: elektromagnetski, akustički, optički, mehanički i inercijski. Poneki koriste više navedenih principa kako bi što točnije sakupili informacije položaja i orijentacije.

Senzori sile i momenta sile mjere silu koju korisnik vrši na uređaj. Takvi uređaji su poprilično intuitivni za rad. Primjerice prilikom manipulacije 3D predmeta u sceni, ukoliko je prisutna veća sila na uređaj, 3D predmet se brže pomiče.

Senzori položaja tijela su kombinacija većeg broja senzora položaja (ponekad i orijentacije) integriranih u odijelo koje kada korisnik nosi dobiva se potpuna slika o

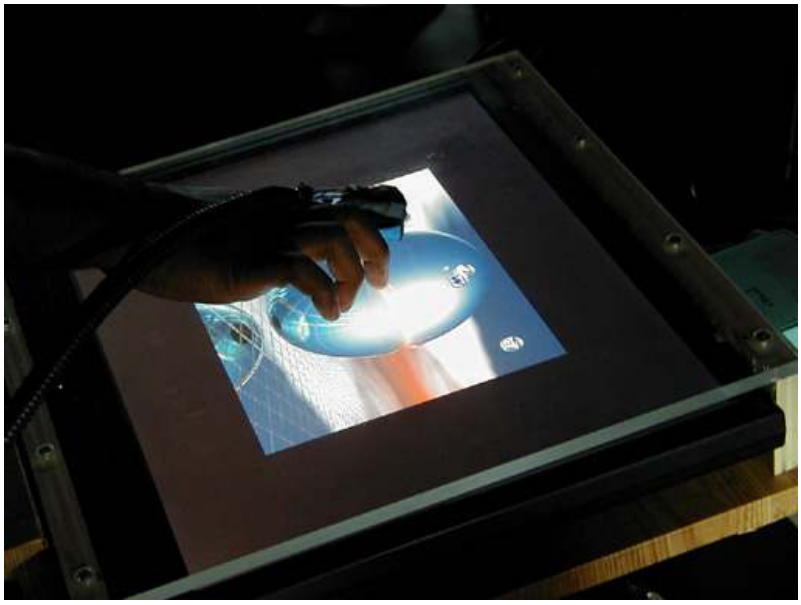
njegovim pokretima. Senzori položaja ruku su integrirani u rukavicama, te mjere kut zakreta zglobova prstiju.

Najpoznatiji uređaj za vizualizaciju virtualne stvarnosti i općenito virtualnih okruženja je zaslona na glavi (*eng. Head Mounted Display*) [1]. Često se na uređaj montiraju senzori za položaj i orijentaciju kako bi korisnik pomakom glave mijenjao smjer gledanja u virtualnom okruženju, te time dobio bolji osjećaj uronjenosti u virtualnu scenu. Slika 3 prikazuje primjer navedenog uređaja.

Slika 3 Primjer zaslona na glavi [3].

Od ostalih uređaja korištenih za vizualizaciju virtualne stvarnosti valja spomenuti stereoskopske ekrane i projekcione sustave. Stereoskopski ekrani omogućuju 3D prikaz i nalik su normalnim ekranima samo što naizmjenično prikazuju sliku za lijevo i desno oko. Korisnik pritom mora nositi odgovarajuće naočale (primjerice zaklopne ili naočale s polarizacijskim staklima), kako bi se odgovarajuća slika projicirala korisniku u odgovarajuće oko. Kod projekcionih sustava moguće su razne konfiguracije projektora koje se mogu koristiti s običnom ili stereo projekcijom. Najpoznatiji sustavi su: CAVE (Slika 4), kod kojih se slika projicira na sve zidove oko korisnika; širokokutne projekcije gdje nekoliko projektora projicira sliku na zaobljenu površinu i virtualni radni stol (Slika 5) gdje slika projicira na površinu koja omogućuje korisniku interakciju s virtualnim okruženjem.

Slika 4 Projekcioni sustav CAVE [4].



Slika 5 Virtualni radni stol [5].

Osim navedenih ulaznih i izlaznih uređaja postoji još niz specijaliziranih uređaja koji su isključivo namijenjeni za određen simulator virtualne stvarnosti. U principu, radi se o replikama stvarnih uređaja za koje se korisnik obučava s time da su na uređaju dodani neki dijelovi prethodno navedenih ulazno/izlaznih jedinica kako bi omogućili interakciju korisnika s virtualnim okruženjem.

1.2. Računalna simulacija

Računalna simulacija je vrlo problematičan dio u petlji virtualne stvarnosti jer je potrebno mnogo računalnih resursa da bi se stvorio sadržaj prema izlaznim jedinicama na temelju podataka iz ulazni jedinica u prihvatljivom vremenu. Ovisno o namjeni i vrsti simulacije prihvatljivo vrijeme može varirati. U simulacijama za obuku se većinom zahtijeva da vrijeme koraka računalne simulacije ne smije biti veće iznad propisanog prihvatljivog vremena, dok u simulacijama za zabavu ponekad vrijeme simulacije smije prijeći propisanu granicu, premda to može biti loše iskustvo za korisnika. Stoga ozbiljnije simulacije zahtijevaju i više računalnih resursa.

1.3. Primjene

Glavne primjene virtualne stvarnosti pojavile su se u slijedećim područjima: medicina, vojne primjene, obrazovanje, zadava, dizajn i razvoj i marketing.

Medicina je jedno od najjačih područja primjene virtualne stvarnosti [1]. Koristi se u kirurgiji za obuku i planiranje kirurških zahvata. Moderniji uređaji mogu načiniti 3D prikaz iz medicinskih snimki što omogućuje bolju vizualizaciju. U psihijatriji se virtualna stvarnost koristi za liječenje raznih poremećaja kao što je strah od visina, strah od određenih životinja, te liječenje posttraumatskog stresnog poremećaja. Očekuje se da će u narednim godinama s napretkom tehnologije virtualna stvarnost imati važnu ulogu u ovom području.

Vojne organizacije su najveći ulagač u razvoj virtualne stvarnosti [1]. Mnoge tehnologije virtualne stvarnosti ugrađene su u simulatore raznih vojnih uređaja i vozila, kako bi se korisnici mogli obučavati u kontroliranoj okolini bez opasnosti po život.

Osim u vojne svrhe, obučavanje pomoću tehnologija virtualne stvarnosti se primjenjuje i u drugim područjima. Velik značaj imaju u obuci civilnog pilota, gdje se među ostalom radi situacijska obuka koja pilota stavlja u situaciju (koju je u stvarnosti teško simulirati) kako bi ju što bolje svladao ukoliko se jednog dana zatekne u njoj. Među ostalim područjima koja koriste virtualnu stvarnost za obuku su: antiterorističke jedinice, vatrogasne postrojbe (uvježbavanje specifičnih požara), visokorizična mjesta (kao što su nuklearne elektrane), nepristupačna mjesta (primjerice svemirske letjelice) i drugo.

Industrija zabava ne zaostaje za korištenjem virtualne stvarnosti, stoga postoje velik broj atrakcija u zabavnim parkovima poput Disneylanda. Iako je cijena opreme za realizaciju virtualne stvarnosti još uvijek visoka, smatra se da će se virtualna stvarnost ubrzo moći koristiti u zabavne svrhe i u vlastitom domu.

Dizajn i razvoj koristi tehnologije virtualne stvarnosti kako bi se što brže i jeftinije izradio prototip. Postoji niz alata i specifičnih uređaja koja su proizvedena isključivo za dizajn i razvoj novih proizvoda.

Pošto je tehnologija virtualne stvarnosti još uvijek novitet za značajan broj ljudi i uz to je poprilično zanimljiva, može se koristiti u marketinške svrhe.

2. Terapija izlaganjem

Terapija izlaganjem (*eng. exposure therapy*) je oblik kognitivno-bihevioralne psihoterapije za smanjenje straha i anksioznih reakcija, posebice fobija koja se temelji na postupcima habituacije, odnosno navikavanja i kognitivne disonancije [6]. Habituacija je psihološki proces smanjenja psihofiziološkog odgovora i ponašanja uvjetovanog određenom pobudom nakon ponovljenog izlaganja istom pobudom tijekom vremenskog razdoblja. Kognitivna disonancija je termin iz psihologije koji označava mentalno stanje neugode u kome osoba doživljava iskustvo dva ili više nekompatibilna vjerovanja ili istodobno kognitivno obrađuje više informacija.

Podvrsta terapije izlaganjem je produžena terapija izlaganjem (*eng. prolonged exposure therapy*) koja svojom fleksibilnošću bolje odgovara na potrebe individualnog pacijenta [7]. Namijenjena je liječenju posttraumatskog stresnog poremećaja (PTSP), depresije, anksioznosti i osjećanja ljutnje.

U terapiji izlaganjem terapeut kontinuirano i postepeno izlaže pacijenta sve stresnijim situacijama kako bi se identificirali uzroci pacijentove nelagode i straha. Terapeut razgovara s pacijentom o njegovim iskustvima i doživljajima, te ga pokušava naviknuti na situaciju bez fobičnog ponašanja. Tradicionalna terapija izlaganjem koristi metode zamišljanja stresnih situacija i suočavanja s njima uživo. Nemogućnost angažiranja pacijenta u dovoljnoj mjeri povezana je s lošim uspjehom terapije izlaganja. Moderniji pristup terapiji izlaganjem je pomoću virtualne stvarnosti.

2.1. Terapija izlaganjem pomoću virtualne stvarnosti

Terapija izlaganjem pomoću virtualne stvarnosti očituje se u većem emocionalnom angažmanu pacijenta, lakše se primjenjuje, jednostavnije se njome upravlja i sigurnija je od izlaganja u živo.

Terapija se ostvaruje tako da pacijent i terapeut imaju interakciju s virtualnom stvarnošću, ali na različite načine. Pacijent je uronjen u okolinu virtualne stvarnosti pomoću zaslona na glavi i senzora pokreta koji mu dopuštaju kretanje i interakciju s okolinom. Prirodni oblik interakcije omogućuje osjećaj prisutnosti i kvalitetniji emocionalni angažman pacijenta što

uvelike doprinosi ukupnom uspjehu terapije. Terapeut koristi grafičko sučelje kako bi manipulirao okolinom virtualne stvarnosti u koju je uronjen pacijent. S obzirom na emocionalno stanje u kojem se pacijent trenutno nalazi, terapeut može podešavati okolinu virtualne stvarnosti kako bi za pacijenta bila stresnija.

3. Sustav za nadzor i upravljanje virtualnom scenom

Početak realizacije programske podrške započinje s korisnikom i njegovim zahtjevima. Iz niza sastanaka s naručiteljem ispostavilo se da će korisnik programske podrške biti terapeut bez naprednijeg poznavanja računalnih tehnologija i bez razumijevanja programskih jezika. Poznavanje korisnika i njegovih sposobnosti i znanja uvelike govori kako treba realizirati programsku podršku ili bolje rečeno kako treba oblikovati programsku podršku da bi ju korisnik u konačnici mogao uspješno koristiti. Primarni funkcionalni zahtjev je omogućavanje korisniku upravljanje sadržajem virtualne scene. U suštini to znači da je korisniku omogućeno ubacivanje (i izbacivanje) objekta s određenim svojstvima i ponašanjem u scenu kroz intuitivno korisničko sučelje. Osim zadovoljavanja primarnog zahtjeva prilikom sastanaka zaključilo se da je potrebno poštivati slijedeće dodatne zahtjeve prilikom oblikovanja programske podrške:

- jednostavnost korištenja programske podrške
- fleksibilnost prilikom upravljanja virtualnom scenom
- automatizacija upravljanja virtualnom scenom
- proširivost programske podrške

Prvi dodatni zahtjev proizlazi iz korisnikovih sposobnosti poznavanja računalnih tehnologija, te bi mu upravljanje virtualnom scenom pomoću programske podrške trebalo biti što intuitivnije moguće. To podrazumijeva da se korisničko sučelje koje se koristi za upravljanje virtualnom scenom mora sastojati od jednostavnih, korisniku poznatih kontrola kao što su: tipke, liste i tablice. Razvijanje i stavljanje kompliciranih kontrola u korisničko sučelje s kojima se korisnik nije susretao u prošlosti može korisnika iritirati, a time se i dodatno komplicira razvoj programske podrške. Stoga se naglašava jednostavnost kao jedna od načela prilikom oblikovanja.

Razmatranjem već postojećih sustava za upravljanje virtualnom scenom ispostavilo se da ne pružaju korisniku dovoljno fleksibilnosti u načinu upravljanja virtualnom scenom. Kao primjer može se navesti predefinirana tipka na korisničkom sučelju koja obavlja akcije

ubacivanja i izbacivanja točno određenog objekta iz virtualne scene. Primjer jednog takvog korisničkog sučelja prikazuje slika 6.

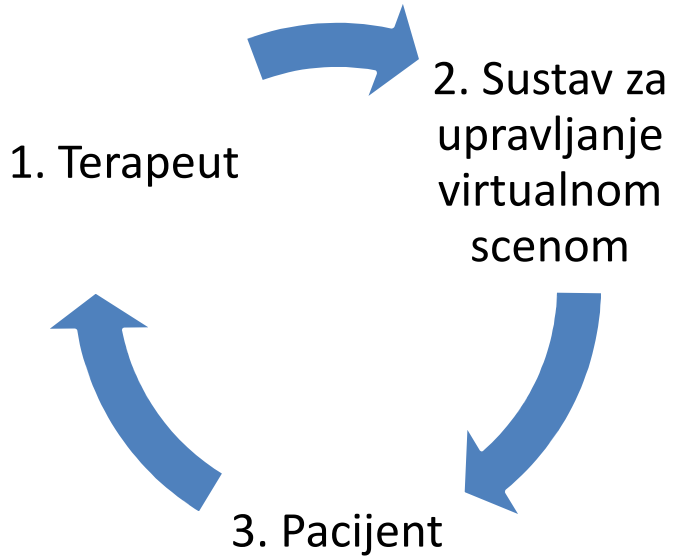
Slika 6 Primjer specifičnog korisničkog sučelja za upravljanje virtualnom scenom [8].

Takav način organizacije korisničkog sučelja omogućava korisniku lako i brzo korištenje programske podrške, međutim pritom se narušava fleksibilnost upravljanja virtualnom scenom. Korisnik može poželjeti staviti neki drugi objekt umjesto nekog postojećeg kojeg više ne želi koristiti u virtualnoj sceni, te se onda zahtijeva dodatno oblikovanje programske podrške, što predstavlja dodatni trošak. Također ukoliko govorimo o terapeutu koji liječi pacijenta, onda bi terapija trebala biti što bolje prilagođena pacijentu, te je time mogućnost ubacivanja specifičnih objekata u virtualnu scene nezaobilazna. Ono što jednom pacijentu predstavlja strah i nelagodu, drugom možda ne predstavlja, stoga ukoliko bi se stalno nadograđivalo korisničko sučelje sa novim tipkama za ubacivanje novih objekata, korisničko sučelje bi postalo prenatrpano tipkama od kojih većina nisu niti potrebna prilikom jedne specifične terapije. Cilj je stoga napraviti fleksibilnu programsku podršku za upravljanje virtualnom scenom, ali isto tako je važno i sačuvati kvalitetna svojstva postojećih aplikacija kao što je lako i brzo korištenje.

Automatizacija je uvijek važan dio poslovnog procesa pošto rasterećuje korisnika koji se tada može posvetiti važnijim dijelom poslovanja. U ovom slučaju govorimo o terapiji, odnosno liječenju pacijenta. Terapeut bi pomoću automatizacije mogao predefinirati ubacivanje objekata s obzirom na vrijeme ili neku drugu akciju, umjesto da on ručno pritiskom na dugme definira željenu akciju. Tako se ne bi trebao brinuti za ubacivanje objekata u scenu, već bi mogao više vremena posvetiti na pacijenta i u kakvom se on stanju nalazi. Pošto prilikom razmatranja postojećih sustava za upravljanjem virtualne scene nije se pronašlo podrške za automatizacijom, trebalo je dodatne inspiracije za realizaciju. Na

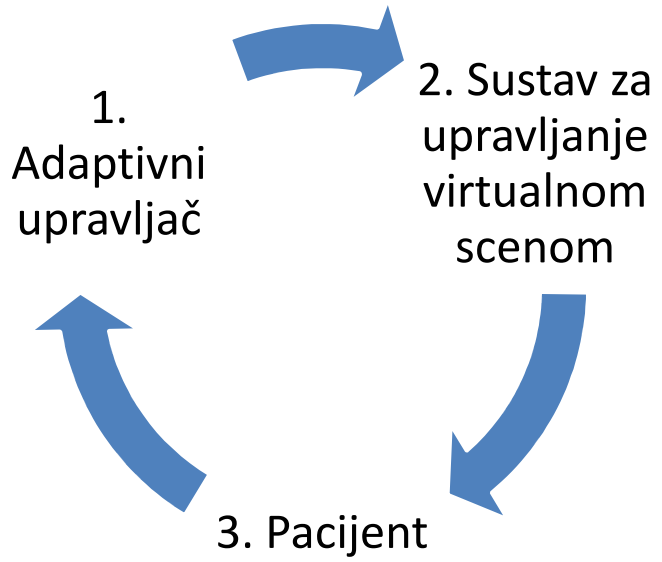
početku poglavlja rečeno je da korisnik ne poznaje programske jezike, stoga nije moguće koristiti tehnike pisanja skriptnih jezika koje bi omogućavale automatizaciju upravljanja virtualnom scenom. Korisniku je stoga potrebno omogućiti posredno programiranje kao što omogućavaju poslovne aplikacije. Kao primjer može se navesti aplikacija za izradu prezentacija (*npr. Microsoft PowerPoint*). Navedena aplikacija omogućuje korisniku animirano dodavanje sadržaja u prezentaciju u ovisnosti o parametrima koje je definirao korisnik. Rezultat toga su ljepše i bogatije prezentacije koje su i odraz korisnikove kreativnosti. Pritom korisnik nije trebao poznavati pisanje programskih jezika da bi omogućio prikaz željenih animacija, već mu je to omogućeno pomoću takozvanog tabličnog programiranja. Inspiracijom takvog načina programiranja omogućit će se automatizacija upravljanja virtualnom scenom.

Osnovna petlja rada sustava za upravljanjem virtualne scene sastoji se od: terapeuta, upravljačkog sustava i pacijenta. Terapeut postavlja početne parametre scene pomoću upravljačkog sustava koji mijenja virtualnu scenu, te tako utječe na virtualnu okolinu pacijenta. Pacijent promjene prihvaća pozitivno ili negativno, te sukladno tome terapeut ponovno mijenja virtualnu scenu. Slika 7 prikazuje navedenu osnovnu petlju rada sustava za upravljanjem virtualne scene.



Slika 7 Osnovna petlja rada sustava za upravljanjem virtualne scene.

Međutim želja naručitelja sustava je da se jednog dana terapeut iz osnovne petlje zamijeni adaptivnim upravljačem. Takav upravljač bi na temelju biološke povratne veze (*eng. biofeedback*) od pacijenta samostalno upravljao virtualnom scenom. Slika 8 prikazuje navedenu napredniju petlju rada sustava za upravljanje virtualnom scenom.



Slika 8 Naprednija petlja rada sustava za upravljanje virtualne scene.

Dosad su bili spomenuti osnovni zahtjevi koji moraju biti ostvareni programskom podrškom. U slijedećim poglavljima biti će rečeno kako su oni ostvareni.

3.1. Koncept rada sustava

Kako je već ranije spomenuto korisnik sustava za nadzor i upravljanje virtualnom scenom će biti terapeut, te će se u daljnjem tekstu on spominjati u ulozi korisnika.

Sustav za nadzor i upravljanje virtualnom scenom je dio većeg sustava koji se osim njega sastoji od sustava u kojem pacijent ima glavnu ulogu, te upravlja iscrtavanjem virtualne scene i omogućuje kretanje istom s obzirom na korisnikove (pacijentove) akcije. Cjelokupni sustav će se stoga sastojati od sustava za iscrtavanje virtualne scene koji će biti realiziran aplikacijom *Pacijent*, dok će sustav za nadzor i upravljanje virtualnom scenom biti realiziran aplikacijom *Terapeut*. Navedene aplikacije će komunicirati putem lokalne

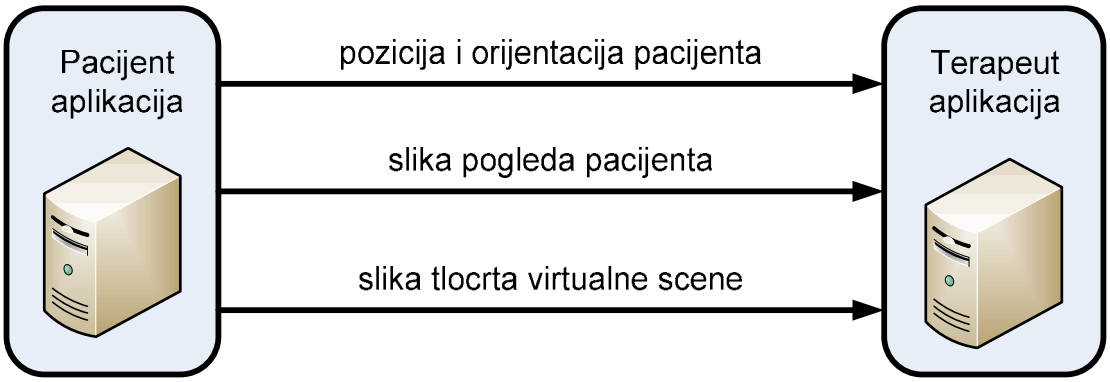
mreže (u teoriji moguće i putem Internet mreže) kako bi razmijenile informacije o virtualnoj sceni i način na koji pacijent utječe na virtualnu scenu. Na slici 9 može se vidjeti fizička arhitektura cjelokupno sustava.

Slika 9 Fizička arhitektura cjelokupnog sustava

Kako bi terapeut aplikacija pružala uvid u virtualnu scenu i položaj pacijenta u njoj, pacijent aplikacija kontinuirano šalje:

- poziciju i orijentaciju pacijenta
- sliku pogleda pacijenta
- sliku tlocrta virtualne scene

Zadaća terapeut aplikacije je da kontinuirano prima navedene podatke, te ih prikazuje terapeutu. Terapeut aplikacija stoga prikazuje sliku pogleda pacijenta u jednom prozoru, dok u drugom prikazuje sliku tlocrta virtualne scene s nadodanim markerom (strelicom) koji označava poziciju i orijentaciju pacijenta u virtualnoj sceni. Pošto pacijent aplikacija šalje 20 puta u sekundi sve navedene podatke, zapravo dobivamo dvije video snimke od 20 slika u sekundi. Slika 10 zornije prikazuje razmjenu pacijentovih podataka između aplikacija.



Slika 10 Kontinuirano slanje pacijentovih podataka

Osim navedenih pacijentovih podataka koji su generirani od strane *Pacijent* aplikacije i šalju se *Terapeut* aplikaciji postoje i podaci koje generira *Terapeut* aplikacija, a prima ih *Pacijent* aplikacija. Takvi podaci su važnijeg prioriteta od kontinuirano slanih podataka kao što je slika pogleda pacijenta, pošto ne smiju biti neisporučeni primatelju (*Pacijent* aplikaciji) i ukoliko se ne mogu procesirati od strane primatelja, onda primatelj mora poslati poruku greške pošiljatelju (*Terapeut* aplikaciji). Također radi konzistentnosti ukoliko je poruka uspješno procesirana, zahtjeva se slanje poruke pošiljatelju o uspješnosti obrade podataka. Idealan model komunikacije s obzirom na gore željeno ponašanje je zahtjev/odgovor. *Terapeut* aplikacija stoga šalje zahtjeve za promjenom virtualne scene i nakon toga dobiva od *Pacijent* aplikacije odgovor. Slika 11 prikazuje spomenuti način komunikacije između *Terapeut* i *Pacijent* aplikacije.

Slika 11 Zahtjev/odgovor način komunikacije između Terapeut i Pacijent aplikacije

Poruke koje predstavljaju zahtjev su dosta specifične s obzirom na željenu radnju, dok odgovor poruke mogu biti općenite, te sadrže osnovne podatke kao što je identifikator, uspješnost provedbe zahtjeva, poruka o pogrešci.

3.2. Komunikacija

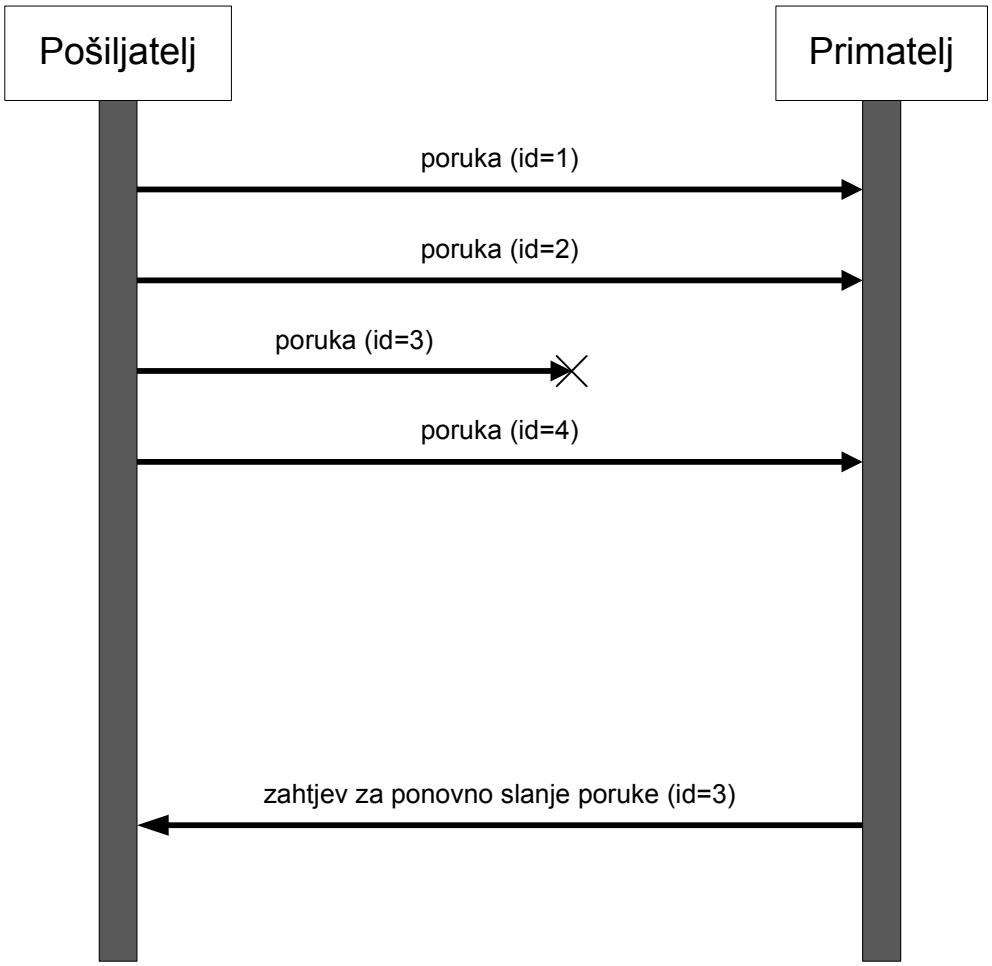
Komunikacije je ostvarena na temelju UDP mrežnog protokola. Karakteristike navedenog protokola su:

- bezkonekcijski transportni protokol
- nepouzdan
- nema kontrole toka

Bezkonekcijski transportni protokoli omogućavaju aplikacijama da šalju poruku (kada se govori UDP-u onda poruke često nazivamo datagramima) bez prethodne uspostave posebnih transportnih kanala ili podatkovnih putova, odnosno bez uspostave konekcije.

Nepouzdan je s obzirom na isporuku poruka. Pošiljatelj može poslati poruku, a da ju primatelj nikad ne primi. UDP neće dostaviti obavijest pošiljatelju da je poruka izgubljena, niti će napraviti ponovno slanje iste poruke. Ukoliko se želi postići pouzdana isporuka poruka na temelju UDP mrežnog protokola, aplikacija mora imati implementiran sustav pouzdane isporuke poruka. Takvi sustavi tada dodjeljuju porukama globalni identifikator na razini aplikacije kojeg stalno uvećavaju prilikom slanja nove poruke. Primatelji pohranjuju identifikator uspješno primljene posljednje poruke, te ukoliko sljedeća uspješno primljena poruka sadrži identifikator koji se razlikuje od očekivanog (primjerice očekuje se da je identifikator za jednu vrijednost veći od zadnje uspješno primljene poruke), onda se lako može saznati koje poruke su izgubljene. Primatelj tada većinom šalje poruku o izgubljenim porukama pošiljatelju, te se odlučuje da li će se poruke ponovno slati (primjerice ukoliko su poruke od veće značajnosti kao što su kontrolne poruke) ili više nisu značajne i ne moraju se ponovno slati (primjerice ukoliko poruke sadrže slike video zapisa). UDP također ne garantira da će poruke biti dostavljene istim redoslijedom kojim su i poslane. Razlog tome je što nije uspostavljen jedinstven podatkovni put i datagrami tada mogu prolaziti različitim putovima i u različito vrijeme dolaziti do primatelja. Naivno slanje zahtjeva za ponovnim slanjem od pošiljatelja primatelju ne garantira dobro ponašanje sustava. Potrebno je dodati vrijeme čekanja prije slanja zahtjeva za ponovno slanje kako bi se dala prilika poruci da dođe i uz malo zakašnjenje. Postoje i drugi načini

rješavanja ovog problema nepouzdanosti. Ovo je jedan od najjednostavnijih za implementaciju. Slika 12 prikazuje implementaciju jednog takvog sustava gdje se poruka s identifikatorom 3 izgubi, te se nakon nekog vremena šalje zahtjev za ponovnim slanjem.



Slika 12 Dijagram primjera zahtjeva za ponovnim slanjem poruke.

Kontrola toka onemogućuje preplavljivanje primatelja porukama. Ukoliko ih primatelj ne stigne obrađivati, poruke se pohranjuju u spremnik pošiljatelja i nakon što se primatelj rastereti, dostavljaju mu se ostale poruke iz spremnika. UDP ne podržava kontrolu toka.

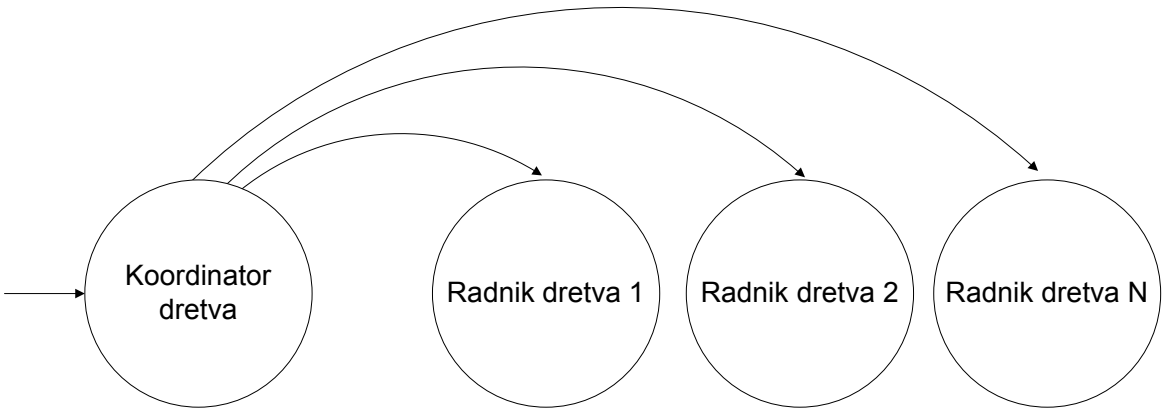
Unatoč svim navedenim nedostacima UDP mrežnog protokola upravo se on koristi za komunikaciju između aplikacija koje zahtijevaju komunikaciju u stvarnom vremenu kao što su aplikacije koje prenose multimedijalni sadržaj (video, zvuk i drugo).

Prilikom realizacije komunikacije u aplikacijama koje periodički razmjenjuju puno podataka potrebno je imati na umu vrijeme koje se troši na:

- zaprimanje poruka
- obradu poruka

Kada se poruka zaprima onda se mora alocirati memorijski prostor u koji će se spremiti podaci. Alociranje memorijskog prostora zahtijeva neko vrijeme, stoga kako bi se eliminiralo to vrijeme prilikom pokretanja aplikacije, unaprijed se stvori nekoliko spremnika isključivo namijenjenih dolaznim porukama. Navedeni spremnici se još nazivaju bazenom spremnika. Aplikacija prilikom zaprimanja poruke tada ne treba tražiti novi memorijski prostor, već samo dohvati jedan slobodni spremnik iz bazena spremnika. Nakon što se završi zaprimanje poruke i spremnik više nije potreban, vraća se u bazen spremnika.

Poruke mogu dolaziti brže nego što ih primatelj može obrađivati. Da bi se riješio navedeni problem potrebno je implementirati koordinator/radnik (*eng. dispatcher/worker*) model rada. U ovom slučaju model će biti ostvaren na razini dretvi. Koordinator dretva cijelo vrijeme osluškuje mrežni promet i zaprima poruke. Nakon što zaprimi poruku prosljeđuje nekoj drugoj dretvi takozvanoj *radnik* na obradu. Kao i sa spremnicima i ovdje se ne treba cijelo vrijeme stvarati nove dretve za obradu, već se može koristiti bazen dretvi koji se stvara na početku aplikacije. Koordinator dretva tako dohvaća slobodnu dretvu i prosljeđuje joj poruku za obradu. Nakon što dretva završi obradu, označava se kao slobodna i stavlja se natrag u bazen dretvi. Slika 13 prikazuje model rada koordinator/radnik.



Slika 13 Model rada koordinator/radnik.

3.3. Biblioteka poruka

U poglavlju 3.1 spomenuto je kako se cjelokupni sustav sastoji od dvije aplikacije: pacijent aplikacije i terapeut aplikacije, koje međusobno komuniciraju. Komunikacija se sastoji od razmjene specifičnih poruka. Kako bi se olakšala implementacija poslovnih logika tih dviju aplikacija razvijena je biblioteka poruka temeljena na objektno orijentiranoj paradigmi.

Korijenski objekt svake poruke ili bolje rečeno omotač sadržaja poruke je klasa `Message`. Razred `Message` u zaglavlju sadrži jedinstven identifikator poruke (*eng. unique message id*) veličine 64 bita i vrstu poruke (*eng. message type*) veličine 16 bita. Jedinstven identifikator služi za identifikaciju poretka poruka, dok tip poruke služi kao oznaka za vrstu obrade koju je potrebno vršiti prilikom njenog primitka. Sadržaj poruke predstavlja apstraktni razred `MessageData`. Na slici 14 može se vidjeti struktura `Message` razreda.

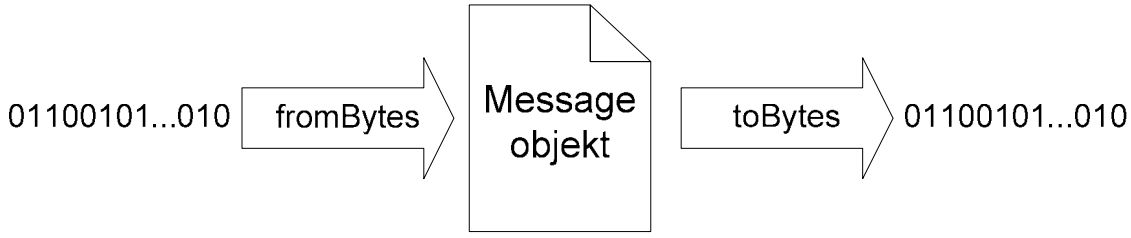
Slika 14 Struktura razreda Message.

Tablica 1 prikazuje moguće vrste poruka. Većina njih su oblika zahtjev/odgovor, gdje zahtjev šalje terapeut aplikacija, a odgovor pacijent aplikacija. Iznimka je zahtjev za deaktivaciju određenog događaja koji mogu poslati obje aplikacije i na njega dobiti odgovor. Razlog omogućavanju *Pacijent* aplikacije da pošalje zahtjev za deaktivacijom je u tome što se time dobiva mogućnost ovisnosti događaja o nekom parametru (kao što je vrijeme postojanja događaja) i omogućavanje međuzavisnosti događaja. Posljednje 3 vrste poruka: pogled pacijenta, slika tlocrta i promjena pacijentovog pogleda u virtualnoj sceni nisu oblika zahtjev/odgovor, već samo pružaju razmjenu podataka od pacijenta.

Tablica 1 Vrste poruka.

Vrsta poruke	Opis	Pošiljatelj
`StartSimulationRequest`	Zahtjev za početkom provedbe simulacije	Terapeut aplikacija
`StartSimulationResponse`	Odgovor s obzirom na zahtjev za početak provedbe simulacije	Pacijent aplikacija
`StopSimulationRequest`	Zahtjev za prekidom izvršavanja simulacije	Terapeut aplikacija
`StopSimulationResponse`	Odgovor s obzirom na zahtjev za prekidom izvršavanja simulacije	Pacijent aplikacija
`ActivateEventRequest`	Zahtjev za aktivacijom određenog događaja	Terapeut aplikacija
`ActivateEventResponse`	Odgovor s obzirom na zahtjev za aktivacijom određenog događaja	Pacijent aplikacija
`DeactivateEventRequest`	Zahtjev za deaktivacijom određenog događaja	Obje aplikacije
`DeactivateEventResponse`	Odgovor s obzirom na zahtjev za deaktivacijom određenog događaja	Obje aplikacije
`PatientViewImage`	Slika pogleda pacijenta u virtualnoj sceni	Pacijent aplikacija
`TopViewImage`	Slika tlocrta terena u virtualnoj sceni	Pacijent aplikacija
`PatientCamera`	Promjena pacijentovog pogleda u virtualnoj sceni	Pacijent aplikacija

Razred `Message` sadrži dvije metode implementirane za pomoć pri ostvarenju komunikacije: `fromBytes` i `toBytes`. Metoda `fromBytes` omogućuje rekonstrukciju instance razreda `Message` iz niza bitova koji se dobivaju putem mreže. Analogno metodi `fromBytes` samo u suprotnom smjeru, metoda `toBytes` omogućuje pretvorbu instance razreda `Message` u niz bitova koji se tada mogu slati putem mreže. Navedene pretvorbe se često nazivaju serijalizacijom, te su prikazane Slika 15.



Slika 15 Serijalizacija Message objekta.

Apstraktni razred `MessageData` predstavlja sadržaj poruke. Razred služi kao sučelje pomoću kojeg se obavlja komunikacija sa specifičnim implementacijama razreda koji sadrže podatke s obzirom na vrstu poruke. S obzirom da postoje specifične implementacije razreda definirane su vrste sadržaja poruke.

Tablica 2 sadrži vrste sadržaja poruke i opis istih.

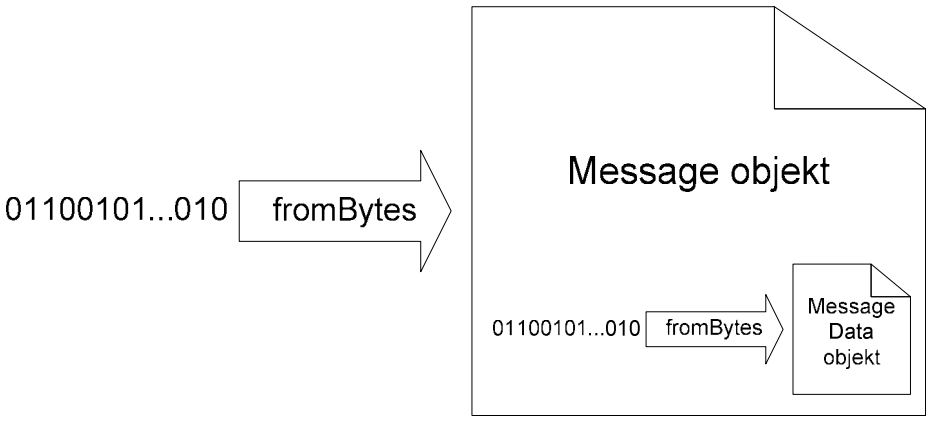
Tablica 2 Vrste sadržaja poruke.

Vrsta sadržaja poruke	Opis
`ImageMessageData`	Sadrži JPEG sliku. `PatientViewImage` i `TopViewImage` koriste ovu vrstu sadržaja.
`PatientCameraData`	Sadrži podatke o pacijentovoj kameri (poziciju i orijentaciju). `PatientCamera` koristi ovu vrstu sadržaja.
`GenericResponseMessageData`	Sadrži zastavicu uspješnosti, identifikator i poruku. Koriste ga sve vrste poruka koje odgovaraju na zahtjev.
`StartSimulationRequestMessageData`	Sadrži naziv terena na kojem se vrši simulacija i početnu poziciju korisnika. `StartSimulationRequest` koristi ovu vrstu sadržaja.
`StopSimulationRequestMessageData`	Sadržaj trenutno ne sadrži podatke. `StopSimulationRequest` koristi ovu vrstu sadržaja.
`DeactivateEventRequestMessageData`	Sadrži naziv događaja za deaktivaciju. `DeactivateEventRequest` koristi ovu vrstu sadržaja.
`XZInsertObjectEventRequestMessageData`	Sadrži parametre translacije (bez Y koordinate), rotacije i skaliranja. `ActivateEventRequest` koristi ovu vrstu sadržaja.
`XYZInsertObjectEventRequestMessageData`	Sadrži parametre translacije, rotacije i skaliranja. `ActivateEventRequest` koristi ovu vrstu sadržaja.

Pretpostavlja se da će broj vrsta sadržaja rasti s daljnjim razvojem sustava. Prvenstveno se pritom misli na dodavanje različitih vrsta ubacivanja događaja u virtualnu scenu. Primjerice događaji koji ovise o drugom događaju ili o udaljenosti od korisnika i slično.

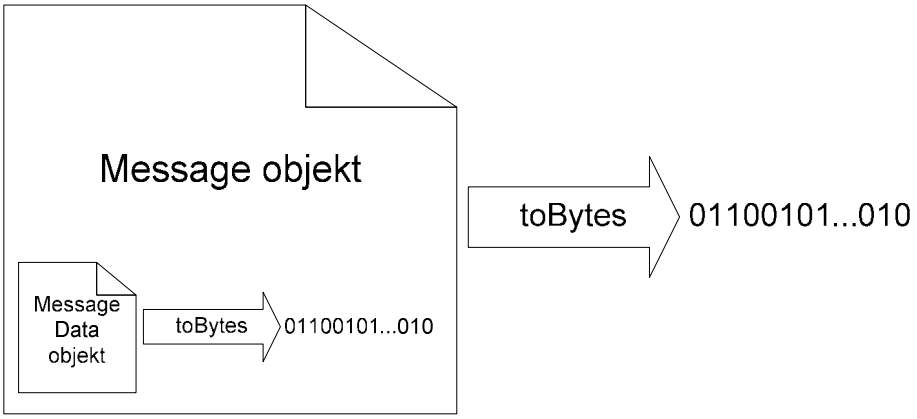
Spomenuto je da razred `Message` sadrži metode `fromBytes` i `toBytes` kako bi obavljao serijalizaciju instance razreda `Message`. Pošto razred `Message` sadrži `MessageData`, navedene metode dodane su i unutar razreda `MessageData`. Metode za serijalizaciju instance razreda `MessageData` se automatski interno pozivaju prilikom poziva metoda za serijalizaciju instance razreda `Message`, te programer koji koristi razred `Message` ne treba brinuti za interne serijalizacije.

Slika 16 prikazuje stvaranje `Message` objekta iz niza bitova. Metoda `fromBytes` od razreda `Message` uzima niz bitova, te vrši rekonstrukciju atributa `Message` objekta. Nakon toga poziva metodu `fromBytes` od razreda `MessageData` kojoj predaje neiskorišteni niz bitova. Metoda `fromBytes` od razreda `MessageData` tada stvara `MessageData` objekt i tada je u potpunosti stvoren `Message` objekt.



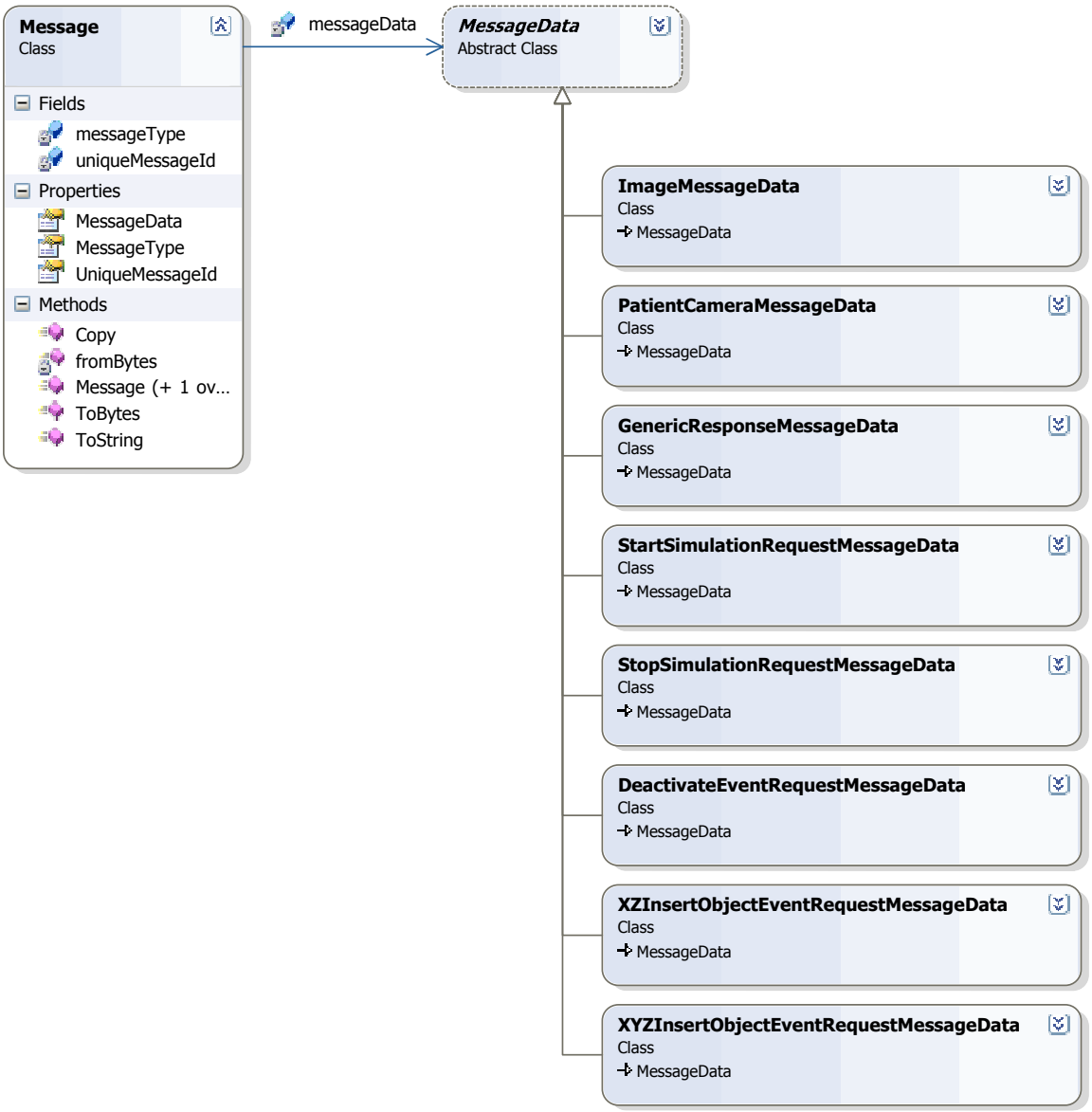
Slika 16 Stvaranje Message objekta iz niza bitova.

Slika 17 prikazuje stvaranje niza bitova iz `Message` objekta. Metoda `toBytes` od razreda `Message` stvara niz bitova tako da na svoj izlazni niz dodaje niz dobiven od metode `toBytes` od razreda `MessageData`.



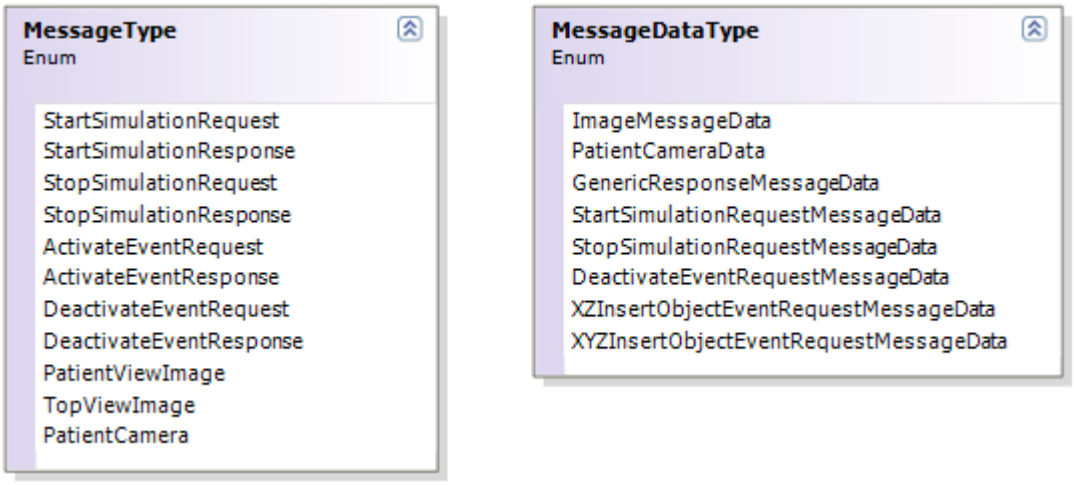
Slika 17 Stvaranje niza bitova iz Message objekta.

Asocijacije i zavisnosti razreda unutar biblioteke poruka mogu se vidjeti na slici 18. Razred `Message` sadrži kao člansku varijablu razred `MessageData` koji može biti implementiran pomoću ostalih razreda čije ime završava s `MessageData`.



Slika 18 Dijagram klasa u biblioteci poruka.

Vrste poruka i vrste sadržaja poruka su sadržane u pobrojanim vrijednostima `MessageType`, odnosno `MessageDataType`. Slika 19 sadrži dijagram pobrojanih vrijednosti u biblioteci poruka.



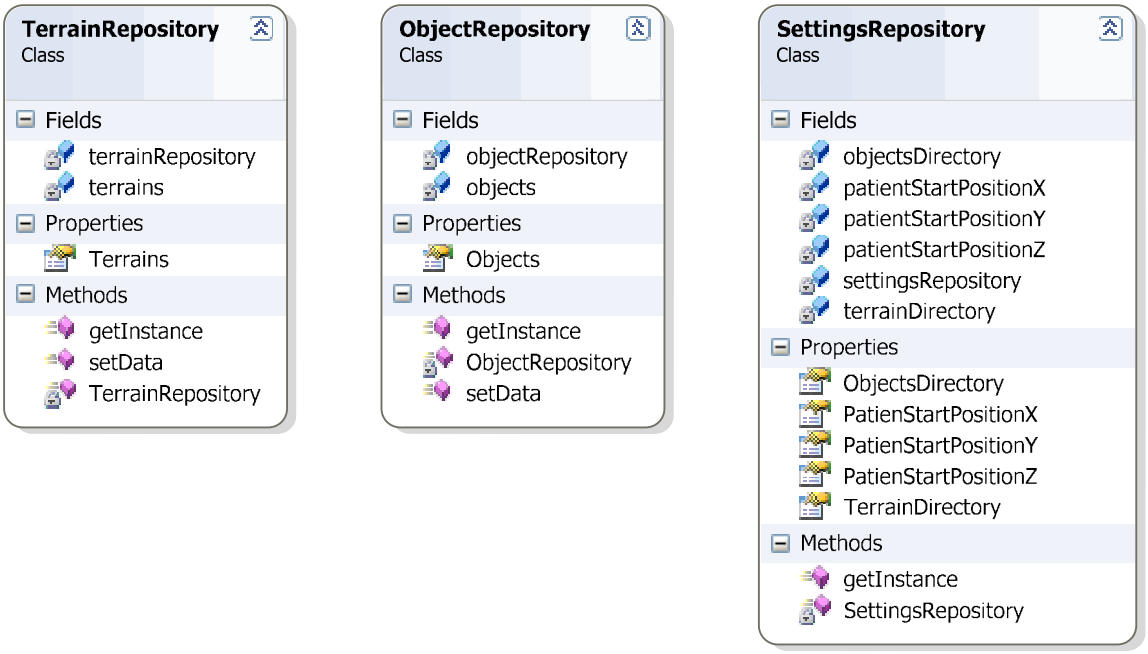
Slika 19 Dijagram pobrojanih vrijednosti u biblioteci poruka.

3.4. Model terapeut aplikacije

Prilikom pokretanja *Terapeut* aplikacija pregledava zadane mape terena i objekata. Imena terena i objekata koji su pronađeni u zadanim mapama spremaju se unutar aplikacije u repozitorije: `TerrainRepository` i `ObjectRepository`. Postavke aplikacije kao što su mape terena i objekata, te početna pozicija pacijenta spremljene su unutar repozitorija `SettingsRepository`.

Svi navedeni repozitoriji implementirani su na temelju Singleton oblikovnog obrasca. Singleton omogućava samo jednu instancu razreda koji ga primjenjuje, te instancu nije potrebno stvoriti `new` operatorom, već se ona dohvaća pozivom `getInstance` metode razreda. Metoda `getInstance` tada provjerava da li već postoji instanca razreda, te ukoliko postoji vraća ju. Ukoliko ne postoji instanca, stvara se nova instanca koja se pohranjuje unutar statičke varijable razreda, te se vraća stvorena nova instanca. Kako se ne bi mogla stvoriti pozivom izvan razreda nova instanca razreda, konstruktor razreda se proglašava privatnim. Singleton time omogućava ponašanje klase nalik ponašanju statičkih klasa, samo što se njene članske varijable i metode ne trebaju proglasiti statičkim (osim naravno varijable instance i metode `getInstance`).

Na slici 20 mogu se vidjeti dijagrami razreda navedenih repozitorija.



Slika 20 Dijagram razreda TerrainRepository, ObjectRepository i SettingsRepository.

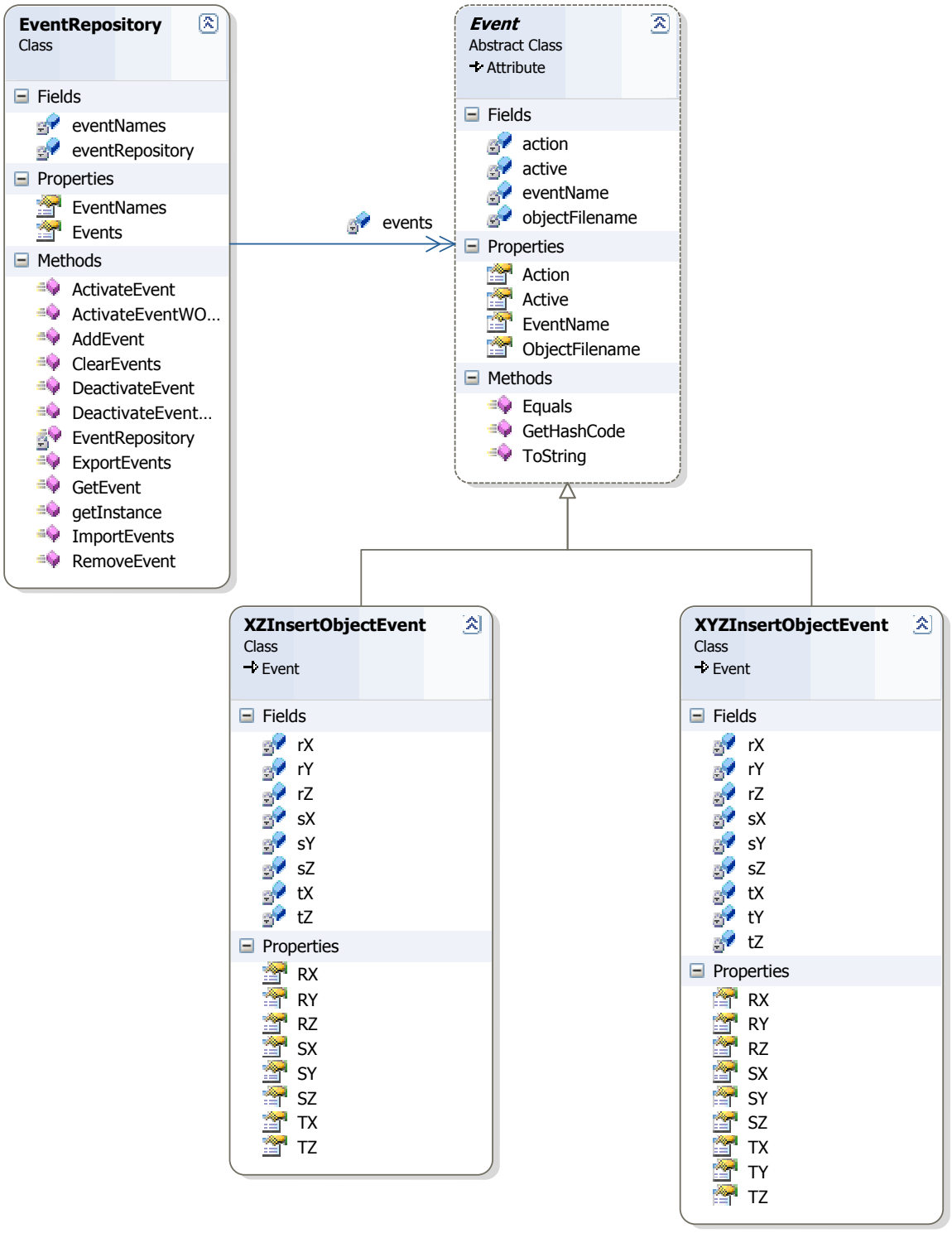
Događaj se definira kao povezanost objekta i načina (akcije) na koji on utječe na virtualnu scenu. Primjerice ubacivanje objekta *cow.obj* na površinu terena bi se omogućilo događajem koji sadrži ime objekta *cow.obj* i akcije `XZInsertObject`.

Apstraktni razred `Event` predstavlja događaj koji sadrži osnovne zajedničke podatke svih događaja:

- ime događaja
- ime objekta
- akcija
- zastavica aktivnosti događaja

Ostali podaci koji su potrebni događaju, odnosno akciji koju događaj sadrži su sadržani unutar specifičnog razreda temeljenog na akciji koji nasljeđuje apstraktni razred `Event`. Trenutno su implentirana dva takva razreda: `XZInsertObjectEvent` i `XYZInsertObjectEvent` koji sadrže dodatne podatke o translaciji, rotaciji i skaliranju.

Za vrijeme izvršavanja terapeut aplikacije događaji su pohranjeni unutar repozitorija `EventRepository`. Navedeni repozitorij također implementira Singleton oblikovni obrazac. Slika 21 prikazuje dijagrame razreda povezanih s događajima.



Slika 21 Dijagrami razreda poveznih s događajima.

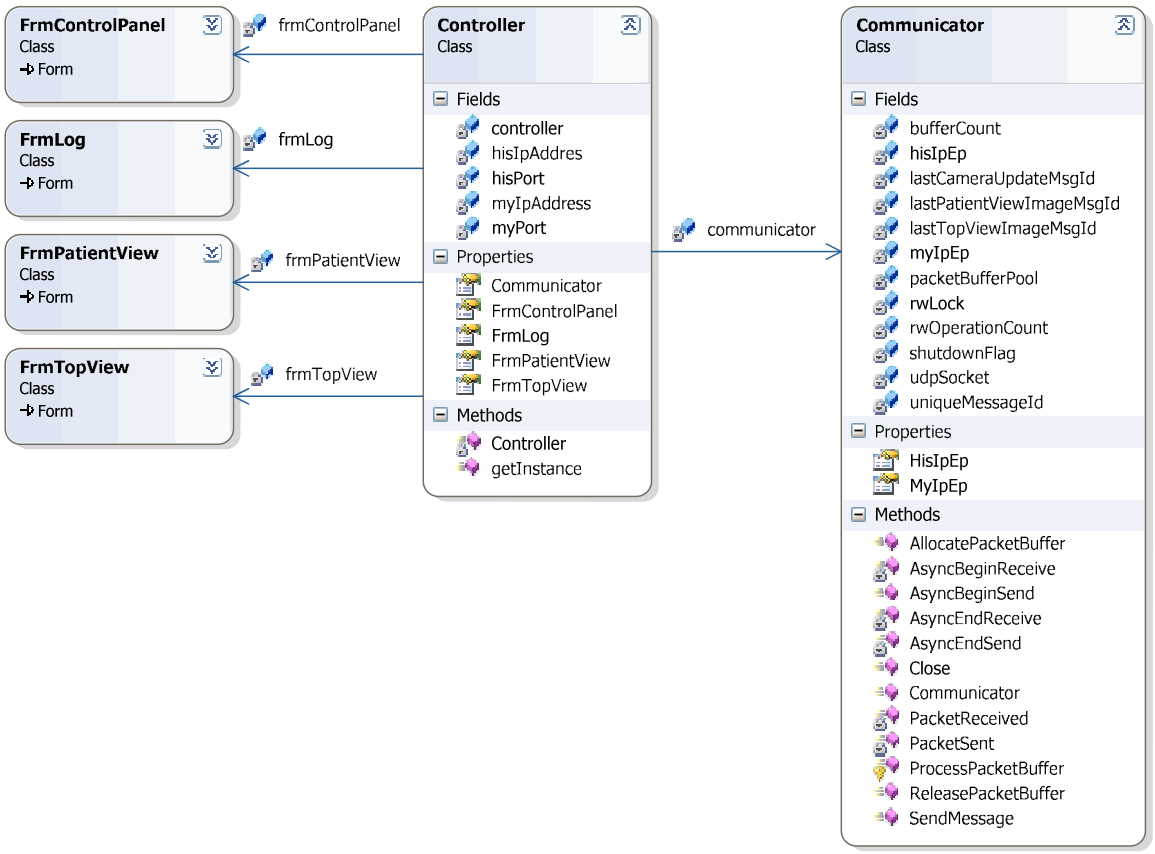
Repozitorij `EventRepository` sadrži metode: `ImportEvents` i `ExportEvents`, kako bi se događaji mogli pohraniti u datoteku i kasnije (nakon prestanka rada *Terapeut*

aplikacije) ponovno uvest u *Terapeut* aplikaciju. Pohranjivanje i uvoz je implementiran na sličan način kao i slanje poruke mrežom. Serijalizacija poruka je ručno pisana kako bi se smanjio mrežni promet, dok je serijalizacija za događaje omogućena s `SerializableAttribute` razredom .NET platforme koja ne pruža optimizaciju kao i ručno pisana serijalizacija, ali to ovdje nije niti potrebno.

Repozitorij je sinkroniziran s pacijent aplikacijom. Primjerice pozivom metode za aktivaciju događaja `ActivateEvent`, osim postavljanja zastavice aktivnosti događaja stvara se poruka za aktivacijom događaja, te se šalje pacijent aplikaciji. Metoda `CreateMessageData` iz razreda `EventRequestMessageDataFactory` omogućuje pretvorbu instance `Event` razreda u instancu `MessageData` koja je potrebna za slanje putem mreže.

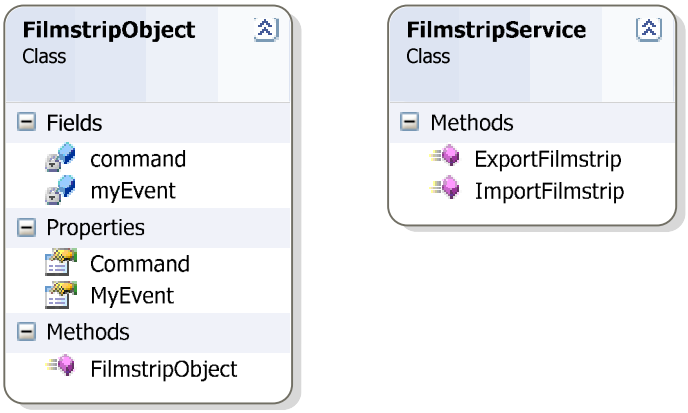
Terapeut aplikacija sadrži centralni upravljač implementiran razredom `Controller`. Upravljač implementira Singleton oblikovni obrazac kako bi se mogao pozvati iz bilo koje metode unutar aplikacije. Prilikom svog stvaranja stvara instancu razreda za komunikaciju (`Communicator`) i instance razreda koji implementiraju korisničko sučelje (`FrmControlPanel`, `FrmLog`, `FrmPatientView` i `FrmTopView`). Stoga ranije spomenuta metoda `ActivateEvent` unutar repozitorija `EventRepository` dohvaća instancu razreda za komunikaciju putem upravljača kako bi mogla poslati poruku pacijent aplikaciji. Slika 22 prikazuje dijagram razreda upravljača i njegovih veza s ostalim razredima.

Kako bi se uspješno koristio razred za komunikaciju potrebno je poznavati dvije metode: `SendMessage` i `ProcessPacketBuffer`. Metoda `SendMessage` prima kao argument instancu razreda `Message` i šalje ju primatelju. Podaci o primatelju (IP adresa i broj vrata) se ne moraju svaki puta navoditi, već su one definirane u `Controller` razredu. Ukoliko će biti daljnjeg razvoja na aplikaciji podatke o primatelju bi svakako trebalo pohraniti u datoteku iz koje se čitaju podaci prilikom pokretanja aplikacije. Metoda `ProcessPacketBuffer` zaprima poruku od pošiljatelja. S obzirom na vrstu poruke metoda određuje kojoj instanci se dalje prosljeđuju podaci poruke. Moguće su optimizacije kao primjerice ukoliko dođe poruka slike pacijenta, ona se ne prosljeđuje ukoliko je njezin identifikator manji od identifikatora posljednje poruke slike pacijenta. Na taj način se odbacuju zakašnjele poruke koje nisu više potrebne. Osim slike pacijenta, slika tlocrta i podaci o položaju i orijentaciji pacijenta su optimizirani na spomenuti način.



Slika 22 Dijagram razreda Controller, Communicator i razreda korisničkih sučelja.

Posljednji dio modela *Terapeut* aplikacije sadrži razrede koji omogućuju realizaciju automatizacije upravljanja virtualnom scenom. Automatizacija je ostvarena pomoću tablice koja u svakom retku sadrži instancu razreda `FilmStripObject` koja sadrži naredbu i događaj. Moguće su 3 vrste naredbe: aktivacija događaja, deaktivacija događaja i čekanje izraženo u sekundama. Instance razreda `FilmStripObject` nemaju svoj repozitorij, već su sadržane u podatkovnom spremniku tablice unutar korisničkog sučelja. Razlog nepostojanja repozitorija je što ne postoji potreba za sinkronizacijom između terapeut aplikacije i pacijent aplikacije za te instance. Kada se provodi automatizacija događaji koji se trebaju aktivirati ili deaktivirati su poslani pacijent aplikaciji iz repozitorija `EventRepository`. Da bi se omogućilo pohranjivanje i učitavanje tablice za automatizaciju napravljen je servis `FilmstripService` koji sadrži metodu za pohranjivanje (`ExportFilmstrip`) i metodu za učitavanje (`ImportFilmstrip`) tablice. Slika 23 prikazuje dijagrame razreda koji omogućuju automatizaciju.



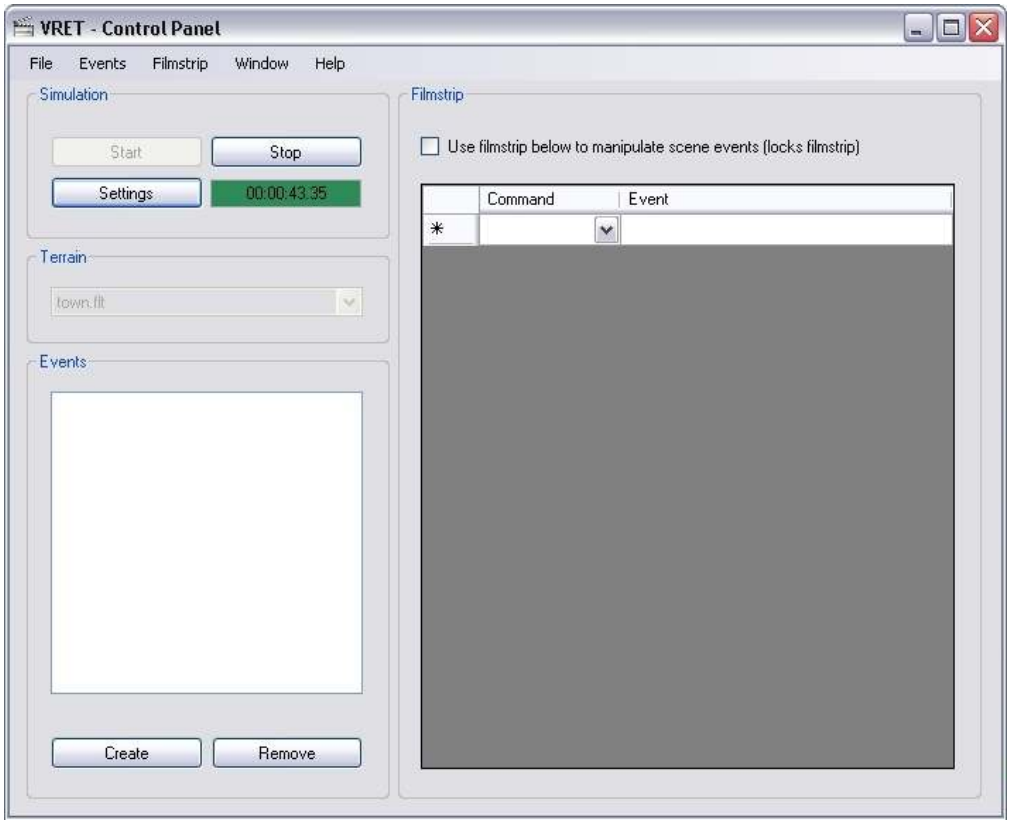
Slika 23 Dijagram razreda FilmstripObject i FilmstripService.

3.5. Korisničko sučelje terapeut aplikacije

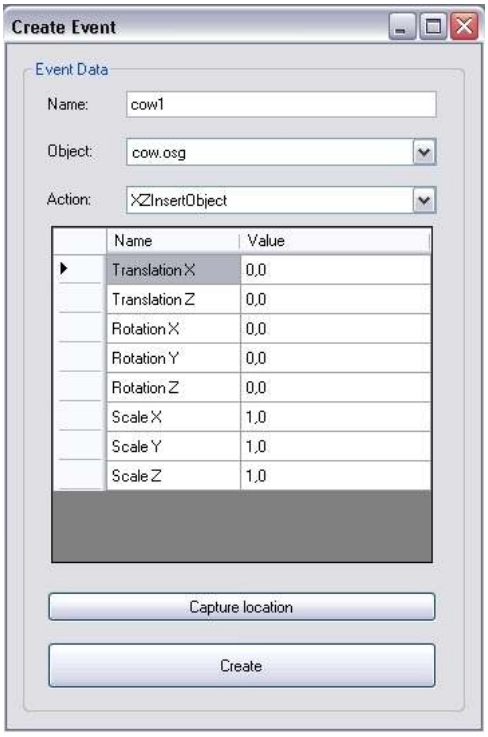
Korisničko sučelje terapeut aplikacije sastoji se od:

- upravljačke ploče
- pogleda na tlocrt virtualne scene
- pogleda pacijenta
- zapisnika

Upravljačka ploča realizirana je `FrmControlPanel` razredom. Sadrži kontrole za upravljanjem simulacije, kontrole za odabir terena, listu događaja i tablicu za automatizaciju. Slika 24 prikazuje upravljaču ploča za vrijeme pokrenute simulacije. Na slici se vidi da je prošlo 43 sekunde i 35 stotinki od pokretanja simulacije s terenom *town.flt* u koji još nije dodan niti jedan događaj. Ukoliko se želi dodati događaj, prvo ga je potrebno stvoriti pritiskom na tipku *Create*. Za stvaranje događaja otvara se novi prozor koji je prikazan slikom 25. Ime događaja mora biti jedinstveno, dok se objekt i akcija biraju iz padajućeg izbornika. Dodatni podaci s obzirom na odabranu akciju su prikazani u tablici, te se njihove vrijednosti mogu mijenjati. Podatke koji opisuju translaciju nije potrebno ručno unositi, već je moguće pritisnuti na tipku *Capture Location* i klikom miša na željenu lokaciju na tlocrtu automatski se unose podaci translacije. Aktivacija i deaktivacija događaja ostvaruje se postavljanjem zastavice događaja u listi događaja.

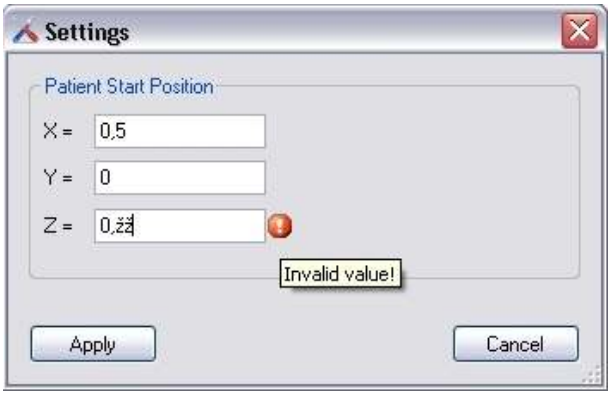


Slika 24 Upravljačka ploča terapeut aplikacije.



Slika 25 Prozor za stvaranje događaja.

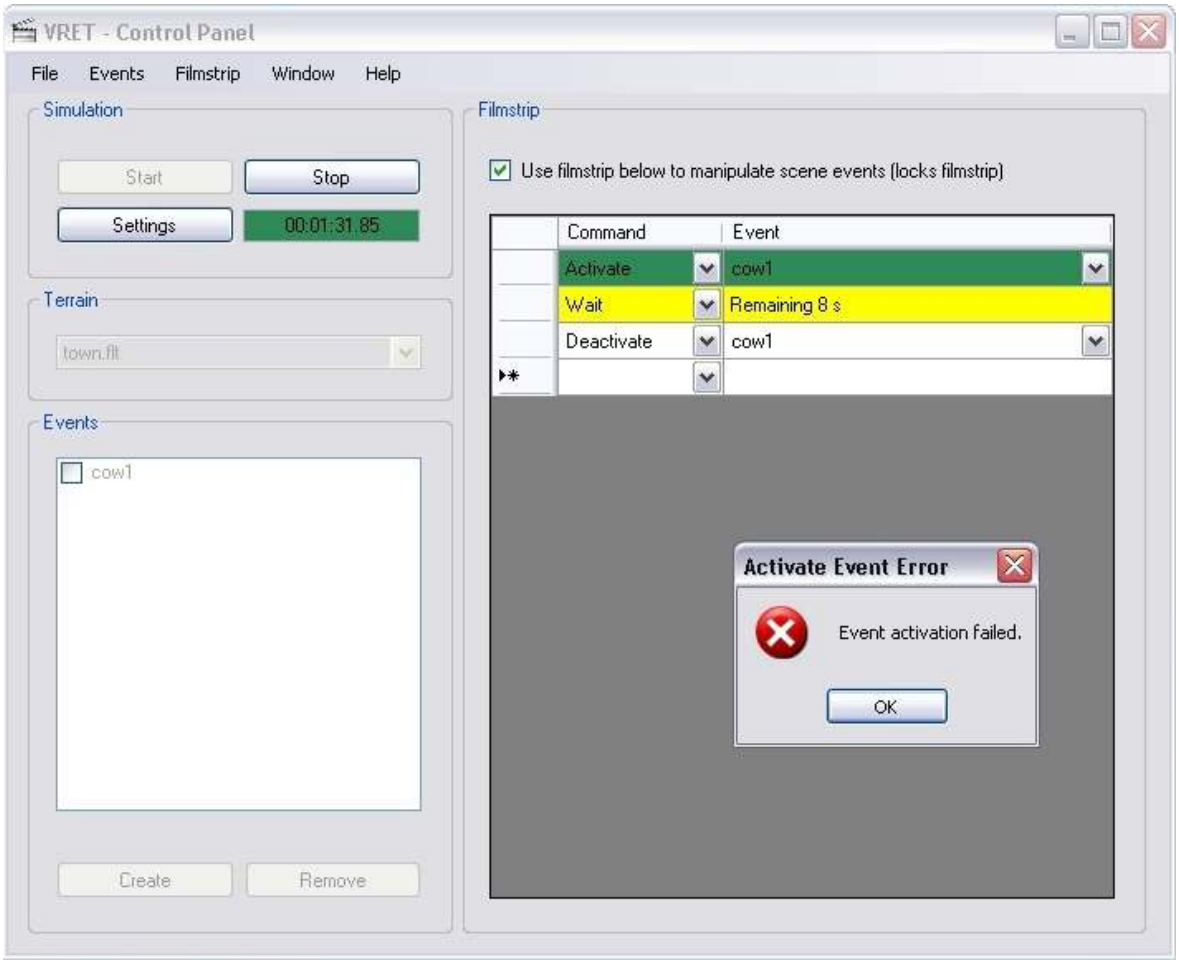
Početna pozicija pacijenta može se postaviti pritiskom na *Settings* tipku koja prikazuje prozor za postavljanje početne pozicije pacijenta. Nad unesenim vrijednostima provodi se validacija gdje x koordinata i z koordinata vrijednosti moraju biti realne vrijednosti od 0 do 1, a y koordinata može biti bilo koja realna vrijednost (ukoliko je ostavljana 0 za y koordinatu, pacijent se postavlja na površinu terena). Slika 26 prikazuje prozor za postavljanje početne pozicije pacijenta.



Slika 26 Prozor za postavljanje početne pozicije pacijenta.

Tablica za automatizaciju puni se odabirom naredbe i događaja s iznimkom ako je naredba *čekanje* onda se u događaj upisuje vrijeme izraženo u sekundama. Svaki redak tablice se izvršava nakon što je redak prije završio. Zelena boja označava da je redak završio izvršavanje, žuta označava da je izvršavanje u tijeku, dok crvena označava da nije bilo moguće pokrenuti izvršavanje s obzirom na stanje u kojem se trenutno nalazi terapeut aplikacija. Moguće su i kasnije dojave da se izvršavanje nije moglo provesti s obzirom na stanje u kojem se nalazi pacijent aplikacija. Pacijent aplikacije će dojaviti grešku porukom terapeut aplikaciji koja će obavijestiti korisnika kao što je prikazano na slici 27. U ovom konkretnom slučaju naredba *aktivacije* umetanja objekta nije se mogla provesti s obzirom da se objekt htio staviti izvan područja terena virtualne scene.

Lista događaja i tablica za automatizaciju su sinkronizirane stoga se u listi događaja mogu vidjeti trenutno aktivirani i deaktivirani događaji koji su upravljani isključivo pomoću automatizacije. Tijekom provedbe automatizacije nije moguće ručno mijenjati listu događaja, bez zaustavljanja izvršavanja automatizacije.



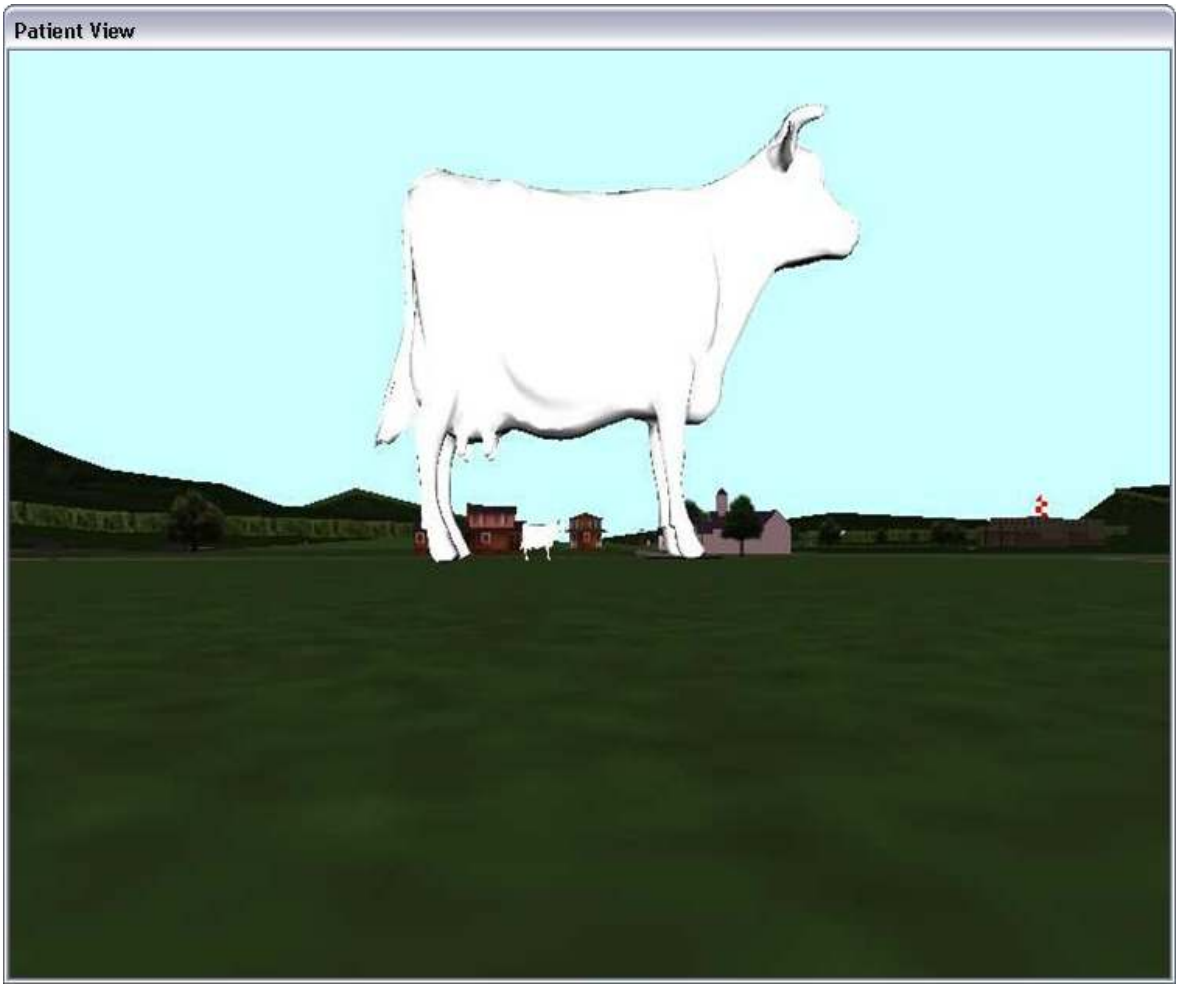
Slika 27 Automatizacija događaja.

Pogled na tlocrt virtualne scene prikazuje sliku tlocrta koju periodički šalje pacijent aplikacija i na nju iscrtava marker koji označuje pacijentovu poziciju i orijentaciju. Implementiran je u razredu `FrmTopView` i ostvaren je kao zaseban prozor kako bi se mogao staviti na zaseban ekran ukoliko računalo ima više ekrana. Razred `FrmTopView` sadrži metode `SetImage` i `SetCameraPosition` koje postavljaju sliku tlocrta virtualne scene, odnosno poziciju i orijentaciju markera koji predstavlja pacijenta. Prilikom primitka `TopViewImage` i `PatientCamera` poruke `ProcessPacketBuffer` metoda unutar `Communicator` razreda poziva `SetImage` i `SetCameraPosition` metode `FrmTopView` razreda putem upravljača kako bi se osvježio pogled. Slika 28 prikazuje pogled na tlocrt virtualne scene na kojem se između ostalog može vidjeti marker (plava strelica) koji označava poziciju i orijentaciju pacijenta i ispred njega bijeli objekt *cow.obj* koji je ranije ubačen i skaliran 10 puta kako bi se vidio na tlocrtu.

Slika 28 Pogled na tlocrt virtualne scene.

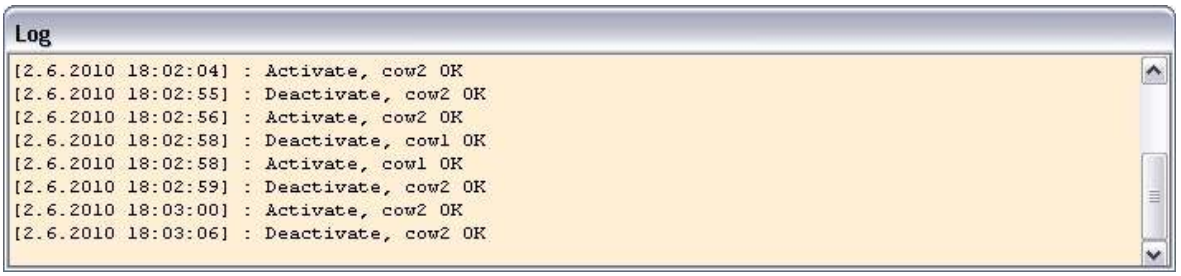
Pogled pacijenta je ostvaren slično kao i pogled na tlocrt virtualne scene. Pogled pacijenta ne zahtijeva dodatne slojeve za markere, tako da se samo slika pacijenta predaje pogledu. Pogled je implementiran `FrmPatientView` razredom i sadrži metodu `SetImage` koja postavlja sliku pacijenta. Slika 29 prikazuje pogledat pacijenta nakon unosa dva objekta.

S obzirom da pacijent aplikacija šalje 20 puta u sekundi slike tlocrta virtualne scene i slike pogleda pacijenta, primanje poruka i iscrtavanje slika zahtijeva dosta računalnih resursa. Preporuka je da se koristi računalo s barem dvije jezgre za potrebe izvođenja terapeut aplikacije kako bi se jedna jezgra koristila za komunikaciju, dok bi se druga jezgra koristila za iscrtavanje korisničkog sučelja.



Slika 29 Pogled pacijenta.

Zapisnik služi kako bi se stvorila povijest radnji napravljenih unutar terapeut aplikacije. Za vrijeme razvoja aplikacije može se koristiti kao dokaz da je korisničko sučelje dobro povezano s modelom aplikacije. Slika 30 prikazuje primjer zapisnika gdje se aktiviraju i deaktiviraju određeni događaji.



Slika 30 Zapisnik.

3.6. Trenutne mogućnosti sustava i daljnji razvoj

Trenutna verzija sustava omogućuje učitavanje proizvoljnog terena u formatu koji je podržan od strane Open Scene Graph programskog sučelja. Dimenzije terena i omjer širine i dužine mogu biti proizvoljne iz razloga što se koordinatni sustav terena u *Terapeut* aplikaciji prikazuje u jediničnom koordinatnom sustavu. Stoga ukoliko je format terena podržan, nema dodatnih potreba za prilagođavanjem sustava ili terena.

Broj događaja koji se može u virtualnu scenu unesti nije ograničen, ali trenutno korisnik (terapeut) može izabrati samo između dvije akcije: ubacivanje na površinu terena i proizvoljno ubacivanje objekata. Pošto je sustav dobro oblikovan, omogućeno je jednostavno dodavanje složenijih akcija kao što su primjerice akcije koje ovise o relativnom položaju objekta i pacijenta (ili nekog drugog objekta), te akcije koje ovise o nekom drugom parametru kao što je vrijeme. Također je omogućeno učitavanje raznih formata objekata koji mogu biti statični ili dinamični (čestični sustavi).

Daljnji razvoj sustava može ići u više smjerova. Jedan je nadograđivanje postojećeg sustava s novim akcijama, te time omogućiti korisniku stvaranje bogatijeg sadržaja u virtualnoj sceni. Drugi može biti razvoj dodatnih komponenti koje komuniciraju s postojećih sustavom kao što je adaptivni upravljač koji olakšava terapeutu provođenje terapije. Bitno je spomenuti da razvoj jednog dijela sustava ne onemogućuje razvoj drugog, već naprotiv svaki novi razvoj čini sustav boljim i bogatijim.

Zaključak

Iako još u ranim fazama razvoja, tehnologija virtualne stvarnosti omogućuje sigurniju i ekonomičniju zamjenu za mnogim radnjama u živo. Mnoge ustanove upravo zato koriste tehnologije virtualne stvarnosti za obuku i usavršavanje. Jedna takva ustanova je bolnica kojoj je cilj što bolje i sigurnije liječiti pacijente koji boluju od posttraumatskog stresnog poremećaja i sličnih fobija. Stoga je ovim radom započet razvoj modernijeg sustava za liječenje takvih pacijenata, kako bi se uklonile mane postojećih (sličnih) sustava i dodale mogućnosti koje još nisu postojale. Rezultat rada je sustav koji podliježe svim zahtjevima propisanim od strane naručitelja kao što su: jednostavnost korištenja, fleksibilnost pri upravljanju virtualnom scenom i automatizacija upravljanja virtualnom scenom, te omogućuje dobru podlogu za nadograđivanje postojećeg i povezivanje postojećeg s nekim drugim (novim) sustavima.

Prilikom testiranja sustava ispostavilo se da je potrebno istodobno izvršavanje više radnji unutar sustava stoga se preporuča korištenje računala s više jezgri ili više procesora i po mogućnosti više ekrana. Idealan broj ekrana za nadzorno upravljački sustav bio bi 3, gdje se jedan koristi za prikaz upravljačke ploče i zapisnika, drugi za prikaz tlocrta virtualne scene s pozicijom i orijentacijom pacijenta, dok bi se treći koristio za prikaz pacijentovog pogleda u virtualnoj sceni.

Literatura

[1] PANDŽIĆ, I. S.: „Virtualna okruženja: računalna grafika u stvarnom vremenu i njene primjene“, Element, Zagreb, 2004.

[2] „Virtual Reality – Wikipedia, the free encyclopedia“, 25.5.2010, *„Virtual Reality“*, http://en.wikipedia.org/wiki/Virtual_reality, 27.5.2010.

[3] „Head Mounted Displays (HMDs) : 3D Stereo Technology: Is it Ready for Prime Time?“, 2.5.2005., *„Head Mounted Displays (HMDs)“*, http://www.tomshardware.com/reviews/3d-stereo-technology,1023-4.html, 20.5.2010.

[4] „Cave Automatic Virtual Environment – Wikipedia, the free encyclopedia“, 18.5.2005., *„Cave Automatic Virtual Environment“*, http://en.wikipedia.org/wiki/Cave_Automatic_Virtual_Environment, 20.5.2010.

[5] KIM, Y. S., KESAVADAS, T.: „UB Virtual Reality Laboratory“, 3.8.2006., *„Fingertip Digitizer: Applying Haptics and Biomechanics to Tactile Input Technology“*, http://www.vrlab.buffalo.edu/projects_haptics/fingertip_digitizer/main.html, 4.6.2010.

[6] ĆOSIĆ, K., HORVAT, M.: „Generator pobuda za estimaciju emocionalnih stanja u interaktivnim simulacijskim sustavima“, LISS, FER, Zagreb

[7] „Prolonged exposure therapy – Wikipedia, the free encyclopedia“, 21.5.2010., *„Prolonged exposure therapy“*, http://en.wikipedia.org/wiki/Prolonged_exposure_therapy, 27.5.2010.

[8] POPOVIĆ, S., ĐUKIĆ, Z., KRIŽEK, V., KUKOLJA, D., ĆOSIĆ K., SLAMNIĆ, M.: „Control of Emotional Characteristics of Stimuli in Virtual Reality Based Psychotherapy“, FER, Zagreb, 2007

Naslov: Interaktivno upravljanje sadržajem virtualne scene – nadzorno upravljački sustav

Sažetak: Ovaj rad opisuje sustav koji primjenjuje tehnologiju virtualne stvarnosti kako bi omogućio bolje i sigurnije liječenje pacijenata koji boluju od posttraumatskog stresnog poremećaja i sličnih fobija pomoću terapije izlaganjem. Sustavom se žele riješiti problemi postojećih sustava koji omogućuju terapiju izlaganjem pomoću virtualne stvarnosti kao što je fleksibilnost u upravljanju virtualnom scenom, te se uvode nove mogućnosti kao što je automatizacija upravljanja virtualnom scenom. Iako je sustav u funkcionalnom stanju, nalazi se tek u početnim fazama vizionarskog sustava koji ne zahtijeva ili jako malo zahtijeva interakciju od strane terapeuta.

Ključne riječi: interaktivno upravljanje sadržajem virtualne scene, nadzorno upravljački sustav, terapeut, pacijent, posttraumatski stresni poremećaj, PTSP, fobije, virtualna stvarnost, terapija izlaganjem

Title: Interactive virtual scene content managment – supervisory control system

Abstract: This paper describes a system which applies virtual reality technology to provide a better and safer treatment of patients suffering from post-traumatic stress disorder and similar phobias using exposure therapy. The system is looking to solve the problems of existing systems that use virtual reality for exposure therapy such as flexibility in managing the virtual scene, as well as providing new features such as automatization of management of the virtual scene. Although the system is in working order, it is only in initial stages of a visionary system that requires no, or very little interaction by the therapist.

Keywords: interactive virtual scene content management, supervisory and control system, therapist, patient, post-traumatic stress disorder, PTSD, phobias, virtual reality, exposure therapy