Sveučilište u Zagrebu

Fakultet organizacije i informatike Varaždin

Marko Škvorc

Razvoj naprednih grafičkih korisničkih sučelja na mobilnim uređajima

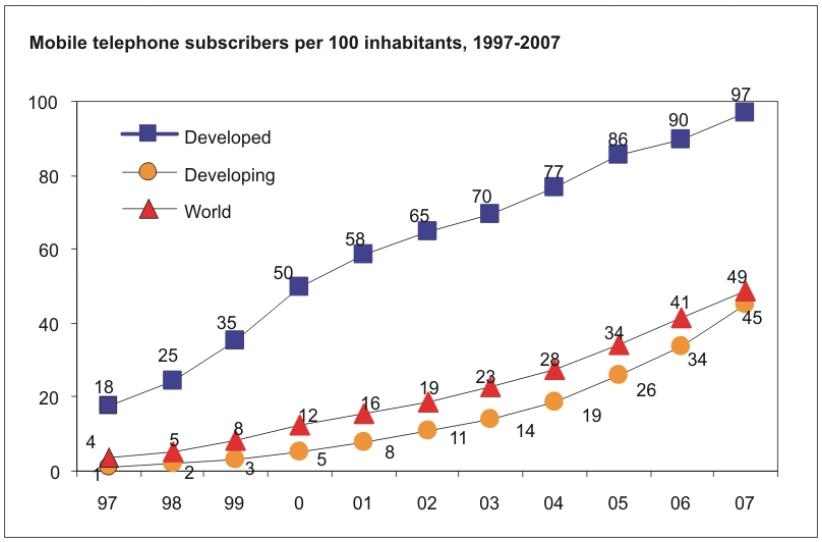
Varaždin, 2010.

Ovaj rad je izrađen na Fakultetu organizacije i informatike u Varaždinu pod vodstvom prof. dr. sc. Nevena Vrčeka i predan na natječaj za dodjelu Dekanove nagrade u akademskoj godini 2009./2010.

Sadržaj

1. Uvod

Računalna grafika prisutna je na osobnim računalima prisutna već dugi niz godina, gotovo od trenutka kada se kao izlazna jedinica počeo koristiti monitor. Međutim, na mobilnim uređajima situacija je malo drugačija. Tek posljednjih desetak godina (od kraja 20. stoljeća pa na dalje), nakon što je započeo nagli porast broja mobilnih uređaja i nagli razvoja njihovih mogućnosti (slika 1.1.), prvenstveno pojavom ekrana u boji, računalna grafika se seli i na mobilne uređaje. U početku su mobilni uređaji doslovno bili mobilni telefoni (mobiteli) i nije bilo potrebe, a niti tehničkih mogućnosti, za izvođenje složenih grafičkih zadataka. Danas, međutim, se sve više nazivaju pametnim telefonima ili malim računalima, jer to zapravo i jesu. Procesorska snaga i memorijski kapacitet današnjih pametnih telefona je u rangu osobnih računala prije desetak godina.



Slika 1.1. Broj mobilnih telefona na 100 stanovnika [ITU, ICT developments over time, developed versus developing countries].

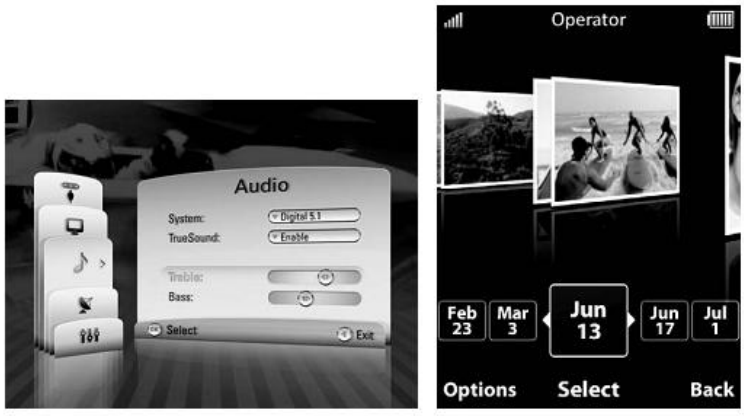
Postepenim dodavanjem dodatnih mogućnosti poput imenika, kalkulatora, kalendara i podrške za SMS poruke [Fling, 2009., str 6.], javila se potreba za prilagodbom samog fizičkog izgleda što je rezultiralo pojavom ekrana na mobilnim uređajima. S vremenom su rasle dimenzije ekrana, ali i kvaliteta prikaza. U uporabu su ušli i zasloni u boji, čija rezolucija se neprestano povećava, pa tako danas nije ništa posebno imati rezoluciju koja je standardna na 15" monitoru. S obzirom na mogućnosti koje pruža takav zaslon, mobilni telefoni su dobili sasvim novu dimenziju. Danas su to uređaji za fotografiranje, pregledavanje slika i video isječaka,

slušanje glazbe, pregled internet stranica, igranje računalnih igara [Fling, 2009., str. 7], zapravo za gotovo sve aktivnosti koje je moguće obaviti i na osobnom računalu. Proporcionalno s porastom broja mogućnosti koje mobilni uređaji pružaju, povećavala se i kompleksnost njihovog korištenja. Veza između mogućnosti mobilnog uređaja i korisnika je korisničko sučelje i upravo je svrha korisničkog sučelja olakšati korištenje mobilnim uređajem.

Tema ovog rada vezana je uz dizajn takvog korisničkog sučelja s naglaskom na njegov grafički izgled koji je također s godinama sve napredniji i sve sličniji onom na koji smo navikli na osobnim računalima. Razvoj jednog takvog grafičkog korisničkog sučelja nije jednostavan zadatak, jer osim programerskog znanja zahtijeva i dobro poznavanje matematike, posebno trigonometrije, te dizajnersko umijeće. U ovom će radu biti prikazani razvoj funkcionalno jednostavnog, ali grafički složenog korisničkog sučelja korištenjem OpenGL ES grafičkog standarda.

2. Ciljevi rada

Današnji mobilni uređaji, pogotovo oni koji pripadaju kategoriji pametnih telefona, imaju prilično veliku procesorsku snagu i brzu memoriju. Zasebni grafički čipovi više nisu rijetkost u mobilnim uređajima, pa stoga ne čudi činjenica kako se mobilni uređaji sve više koriste i za igranje raznih grafički vrlo zahtjevnih igara, osobito među mlađom populacijom. Cilj ovog rada je iskoristiti ovakve mogućnosti i izraditi grafički bogatu aplikaciju za mobilni uređaj koja omogućuje pristup osnovnim funkcionalnostima koje mobilni uređaj pruža (poput pregleda SMS poruka, Email poruka, kontakata i sl.). Primjeri takvih aplikacija prikazani su na slici 2.1. No, naglasak nije na samom programerskom zadatku, već na povezivanju matematike i računalne grafike. Već je ranije navedeno kako je za realizaciju jednog takvog projekta, osim dobrih programerskih vještina, potrebno i dobro znanje matematike, posebno geometrije, što nam omogućuje da teorijska znanja primijenimo u praksi na konkretnom primjeru. Prikaz 2D objekata na zaslonu mobilnog uređaja, ne razliku se previše od crtanja po papiru u zadanom koordinatnom sustavu. Stvari se kompliciraju prelaskom u treću dimenziju, jer je objekte iz 3D prostora potrebno prikazati u 2D prostoru ekrana mobilnog uređaja [Malizia, 2006., str. 13]. Ovaj problem rješavaju brojne grafičke biblioteke, kao što su OpenGL i DirectX, koje sadrže gotove matrične izračune transformacija, a programer je zadužen za oblikovanje objekata i njihov smještaj u zadani prostor. Više o upotrebi matrica u računalnoj grafici u [Ducker, 2000.].



Slika 2.1. Primjeri 3D grafičkog korisničkog sučelja na mobilnom uređaju [Pulli, 2007., str 16]

3. Materijal i metode

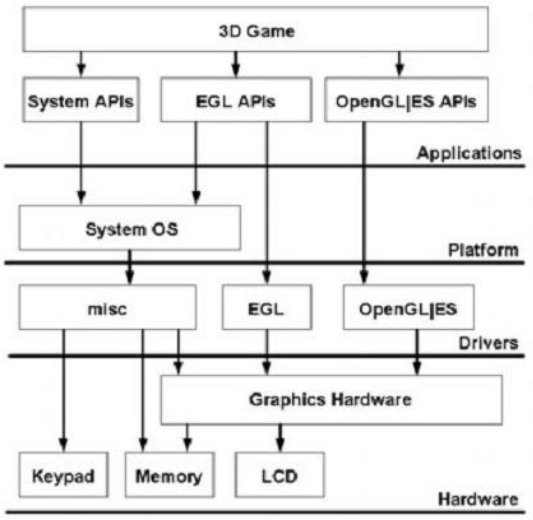
Razvoj bilo koje vrste aplikacije za mobilni uređaj može se podijeliti u nekoliko faza od kojih je prva i osnovna odabir platforme na kojoj će se aplikacija izvoditi. Naime, postoji nekoliko različitih operacijskih sustava za mobilne uređaje koji međusobno nisu kompatibilni, što znači da se ista aplikacija, tj. ista izvršna datoteka neke aplikacije, ne može pokrenuti na različitim operacijskim sustavima. Taj problem u većini slučajeva nemaju aplikacije koje se izvršavaju u internet pregledniku, jer za isti postoje odgovarajući dodaci (engl. plugins) koji su dostupni za sve platforme. Imajući to u vidu, važno je odabrati i odgovarajuću tehnologiju za razvoj.

S obzirom da se ovaj rad temelji na izradi grafičkog korisničkog sučelja posebna pažnja je posvećena odabiru odgovarajućeg grafičkog standarda, a programski jezik je stavljen u drugi plan. Za potrebe ovog rada odabran je OpenGL ES [Munchi, 2008.], inačica OpenGL-a [Segal, 2004] namijenjena izvođenju na mobilnim i drugim uređajima slabije snage u odnosu na osobna računala. "OpenGL ES je programsko sučelje prema grafičkom hardveru. Sučelje se sastoji od niza procedura i funkcija koje programeru omogućuju opisivanje objekata s ciljem izrade grafičkih slika visoke kvalitete, pogotovo slika u boji kao sastavnih dijelova 3D objekata." [Munchi, 2008, str. 1]. OpenGL ES, a i njegova puna inačica, je neovisan o platformi i hardveru, što bi značilo da se može izvršavati na bilo kojem uređaju, no kako je riječ o grafičkom standardu, ne postoji podrška za povezivanje na periferne jedinice poput tipkovnice, zvučnika i sl. Stoga se OpenGL ES u većini slučajeva učahuruje unutar standardnog programa, primjerice nekog C++ programa koji je zadužen za inicijalizaciju, obradu pogrešaka, komunikaciju s okolinom i sl., a OpenGL ES naredbe služe isključivo za prikaz grafičkih elemenata na zaslonu. Upravo zbog toga se aplikacija, koja koristi OpenGL biblioteke, ne može koristiti na različitim platformama.

3.1. Priprema okoline

S obzirom da OpenGL ES nema pristup resursima uređaja, potrebno je osigurati i odgovarajuću vezu prema izlaznoj jedinici, tj. ekranu na kojem se prikazuje grafika. Ovaj tip veze osigurava EGL, sučelje između OpenGL ES programskog koda i *native platform window* sustava [Astle, 2004., str. 5], tj. razine operacijskog sustava zadužene za komunikaciju sa zaslonom. Iako je i EGL otvoreni standard i neovisan o platformi, moraju postojati određene "dodirne točke" prema *native window* sustavu. Time je uvelike olakšana prilagodba aplikacije

za različite platforme, jer se izmjene u programskom kodu vezanim uz EGL odnose samo na izmjenu parametara, dok su pozivi funkcija jednaki. Na slici 3.1. je prikazan položaj EGL-a u odnosu na sloj operacijskog sustava i sloj aplikacije.



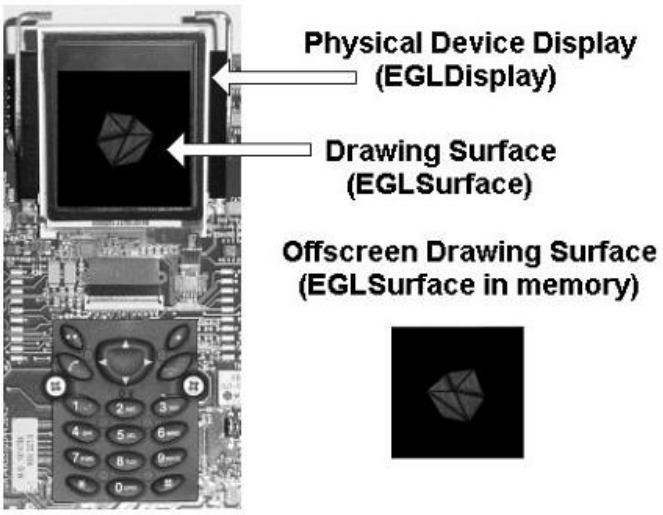
Slika 3.1. EGL kao dio sustava [Astle, 2004., str. 6]

EGL sadrži tri glavne komponente, tj. EGLDisplay, EGLSurface i EGLContext. EGLDisplay je objekt koji predstavlja fizički ekran uređaja. I dok je pristup primarnom ekranu neovisan o platformi, u slučaju da želimo pristupiti nekom drugom ekranu, potrebno je koristiti sistemske pozive specifične za odabranu platformu. Pristup primarnom ekranu uređaja ostvarujemo jednostavnim pozivom funkcije eglGetDisplay, pri čemu kao parametar prosljeđujemo EGL_DEFAULT_DISPLAY [Pulli, 2007., str. 243]. Slijedi inicijalizacija EGL-a pozivom funkcije eglInitialize koja kaj parametar prima pokazivač na ekran, kojeg smo prethodno dobili.

```
eglInitialize(EGLDisplay display, EGLint *major, EGLint *minor)
```

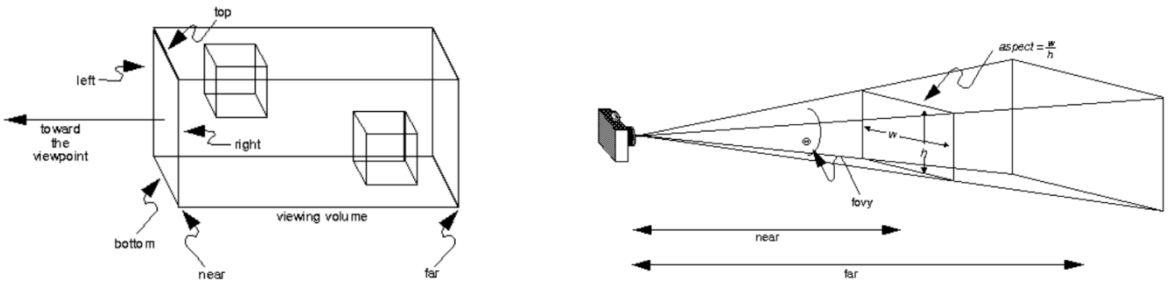
Sljedeći korak je odabir konfiguracije za međuspremnik. Naime, da bi se objekti mogli pravilno iscrtavati potrebno je definirati veličinu polja za pojedinu boju, veličinu pomoćnih međuspremnika i sl. Odabir konfiguracije vrši se pozivom funkcije eglChooseConfig ili eglGetConfigs. Nakon odabira konfiguracije slijedi kreiranje EGLSurface i EGLContext objekata. EGLContext objekt predstavlja kontejner koji pamti interno stanje OpenGL ES koda. U tom su kontejneru spremljene vrijednosti vezane uz teksture, koordinate, vektore, matrice, osvjetljenje, materijal i sl. [Pulli, 2007., str. 252]. EGLSurface predstavlja kontejner u kojem će se prikazati obrađena slika, što je u većini slučajeva zaslon uređaja [Pulli, 2007.,

str. 243]. Odnos između objekata EGLDisplay i EGLSurface prikazan je na slici 3.2. Važno je napomenuti da se za svaki EGLSurface objekt može koristiti više EGLContext objekata i isto tako jedan EGLContext objekt može se koristiti za više EGLSurface objekata. Jednostavnije rečeno, na svakom ekranu uređaja može se prikazati sadržaj koji je unaprijed pripremljen u različitim kontejnerima (međuspremnicima) i sadržaj iz istog kontejnera (međuspremnika) može se prikazati na više ekrana. Detalji o mogućnostima EGL-a mogu se pronaći u [Leech, 2010.].



Slika 3.2. Grafički prikaz EGLDisplay i EGLSurface objekata [Astle, 2004., str. 8]

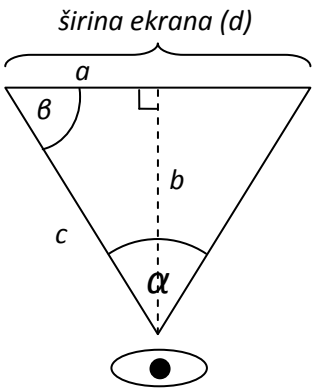
Nakon što su pripremljeni međuspremnici i osiguran pristup ekranu uređaja korištenjem poziva EGL funkcija, slijedi priprema koordinatnog sustava u kojem će se prikazivati objekti. Naime, s obzirom da objekte pozicioniramo u 3D prostoru, a i oni sami su trodimenzionalni potrebno je taj trodimenzionalni sustav pretvoriti u dvodimenzionalni sustav za prikaz na ekranu, tj. potrebno je definirati projekciju. OpenGL definira dva osnovna tipa projekcije: ortogonalnu i perspektivnu [Joubert, 2008., str. 1]. Grafički prikaz projekcija prikazan je na slici 3.3. Glavna značajka ortogonalne ili paralelne projekcije je očuvanje veličine i kutova između objekata bez obzira na njihovu udaljenost od gledatelja. S druge strane, kod perspektivne projekcije, objekti koji su bliže gledatelju, odnosno kameri, su veći, a oni udaljeniji manji, kako i mi percipiramo objekte u stvarnom svijetu [Fosner, 2006.].



Slika 3.3. Grafički prikaz ortogonalne (lijevo) i perspektivne projekcije (desno). [Joubert, 2008., str. 2]

Osim tipa projekcije, potrebno je definirati i poziciju kamere, tj. poziciju s koje korisnik promatra scenu. Pozicioniranje kamere nije ništa drugo nego pomicanje scene, ali u suprotnom smjeru. Primjerice, pomakom kamere prema lijevo za 100 piksela, dobivamo isti pogled kao i pomakom scene prema desno za 100 piksela. Za potrebe ovog rada odabrana je perspektivna projekcija kako bi se postigao pravi 3D doživljaj. Za perspektivnu projekciju je potrebno definirati kut pogleda u smjeru y-osi, omjer širine i visine ekrana te granične vrijednosti na z-osi unutar koji se iscrtavaju objekti. Da bi osigurali vidljivost čitave scene na ekranu potrebno je i dobro pozicionirati kameru, u ovom slučaju odrediti njezinu udaljenost po z-osi od najbliže točke područja crtanja. Kako je i prikazano na slici 3.4., na temelju zadanog kuta i visine ekrana možemo izračunati traženu udaljenost korištenjem sljedećih trigonometrijskih izraza:

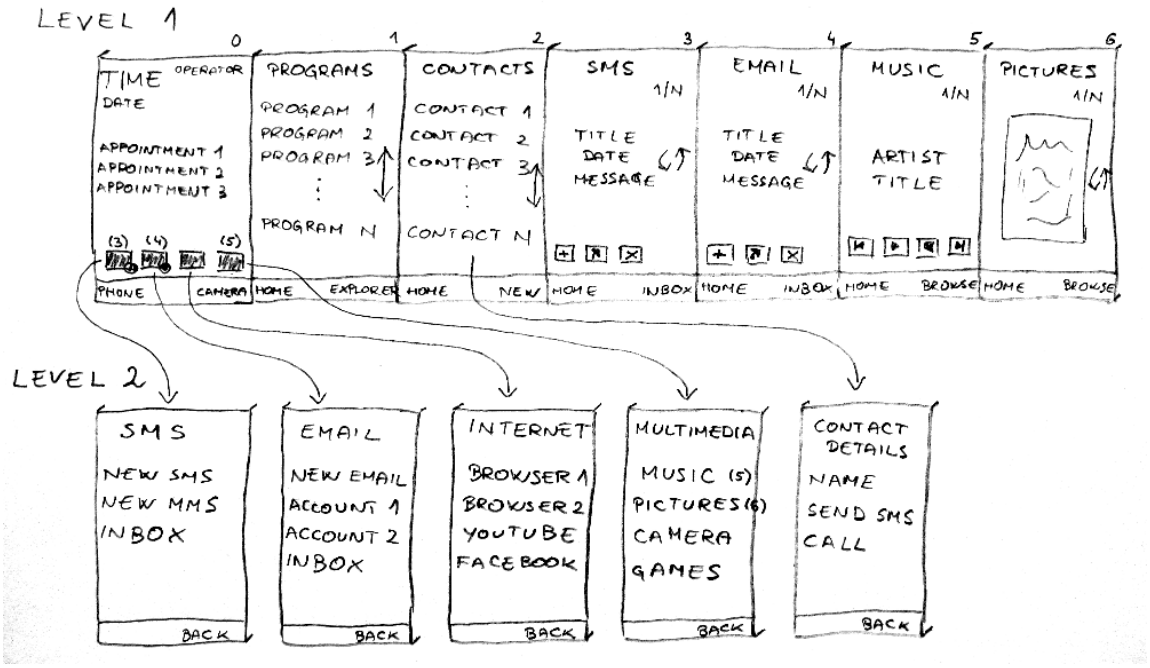
$$sin\beta = \frac{b}{c}, cos\beta = \frac{a}{c}, r = \frac{h}{d} \quad \Rightarrow b = \frac{a \times sin\beta}{cos\beta} \times r$$



Slika 3.4. Grafički prikaz izračuna udaljenosti kamere od područja crtanja

3.2. Skica rješenja

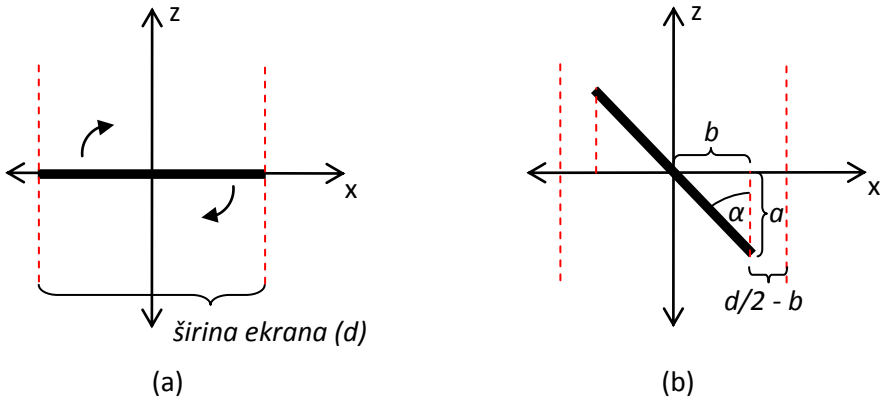
Nakon što je pripremljena okolina za razvoj, može započeti i sam proces razvoja programskog proizvoda. Skica zamišljene aplikacije prikazana je na slici 3.5. Aplikacija se sastoji od sedam ekrana prve razine i pet ekrana druge razine. S obzirom da je naglasak na vizualnom identitetu važno je poštivati osnovne principe dizajna [Ballard, 2007.], od kojih je za potrebe ovog rada zasigurno najznačajniji princip prilagodbe sučelja za korištenje jednom rukom. To prvenstveno znači da su elementi na ekranu, koji zahtijevaju direktnu interakciju s korisnikom (pritisak prsta, povlačenje prsta i sl.) na dohvat palca koji se u većini slučajeva koristi za interakciju s mobilnim telefonom.



Slika 3.5. Skica ekrana zamišljene aplikacije

3.3. Algoritmi animacija

Kako je već i ranije navedeno, cilj ovog rada nije isključivo izraditi programsko rješenje, već prikazati na koji način iskoristiti matematička znanja u računalnoj grafici. Stoga, u nastavku slijedi opis nekoliko osnovnih grafičkih zadataka, uključujući skice i pseudoalgoritme, korištenjem jednostavnih matematičkih formula.



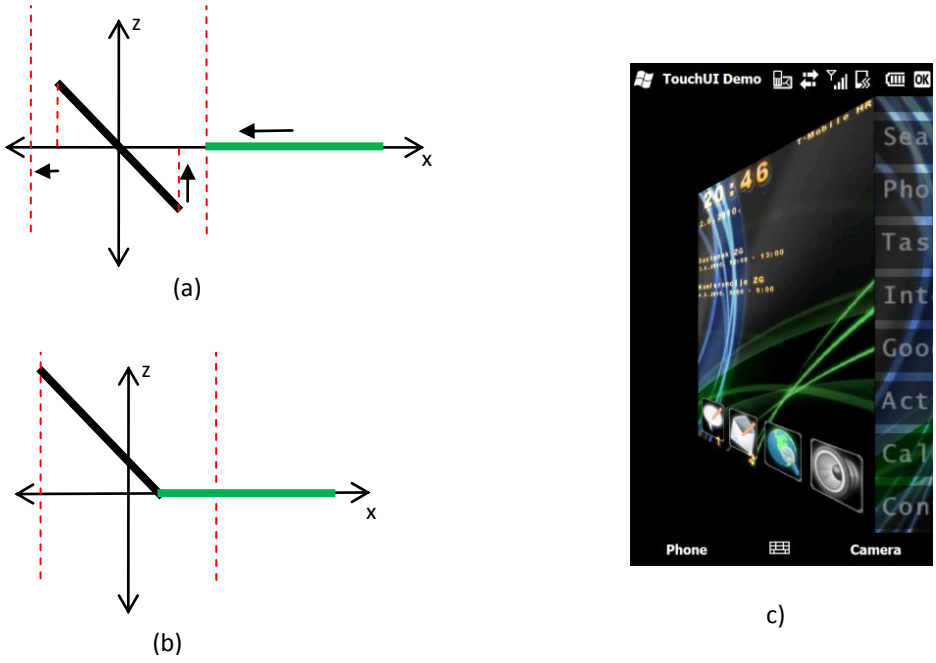
Slika 3.6. Prikaz pripreme animacije u zx-koordinatnom sustavu. Trenutni ekran se rotira oko y-osi za kut α (a) i translatira po z-osi za *a* i x-osi za *d/2 - b* (b).

Prvi u nizu grafičkih zadataka je prikazati animaciju prilikom promjene ekrana prve razine. Riječ je o jednostavnoj animaciji koja uključuje klizanje trenutnog ekrana po zamišljenoj stranici na lijevom, odnosno desnom, rubu ekrana i klizanje novo odabranog ekrana po zamišljenoj stranici na prednjoj strani ekrana (slika 3.6 i 3.7.). Animacija se izvodi u dva koraka. Najprije se rotira trenutni ekran, a nakon toga mu se s lijeve, odnosno desne strane pridružuje sljedeći ekran. Osim rotacije, trenutni ekran je potrebno i translatirati od kamere kako bi njegov rub bliži kameri zadržao z koordinatu te translatirati lijevo, odnosno desno, kako bi rub koji je udaljeniji od kamere zadržao x koordinatu. Da bi se animacija uspješno izvršila potrebno je zadati kut rotacije α u rasponu od 0 do 90 stupnjeva, dok se iznosi za translaciju računaju prema sljedećim formulama.

$$sin\alpha = \frac{a}{d/2} \Rightarrow a = \frac{sin\alpha \times d}{2}$$

$$cos\alpha = \frac{b}{d/2} \Rightarrow b = \frac{cos\alpha \times d}{2}$$

Kao što je vidljivo na slici 3.6. (b) trenutni ekran se najprije rotira za kut α, nakon čega se translatira po z-osi za vrijednost *a* i x-osi za vrijednost *d/2-b*. Sljedeći ekran se samo translatira po x-osi za dvostruku vrijednost translacije trenutnog ekrana po x-osi, tj. za vrijednost *d - 2b*. Translacije i konačne pozicije ekrana grafički su prikazane na slici 3.7.



Slika 3.7. Prikaz izvođenja animacije u zx-koordinatnom sustavu. (a) Translacija trenutnog i sljedećeg ekrana. (b) Pozicije trenutnog i sljedećeg ekrana nakon provedenih transformacija. (c) Isječak animacije s mobilnog uređaja.

Preostaje još prikazati pseudoalgoritam za opisanu animaciju, pri čemu je važno naglasiti kako je bitno slijediti redoslijed transformacija, jer se ekran treba rotirati oko ishodišta, tj. prije translacija, u protivnom dobivamo drugačiji rezultat.

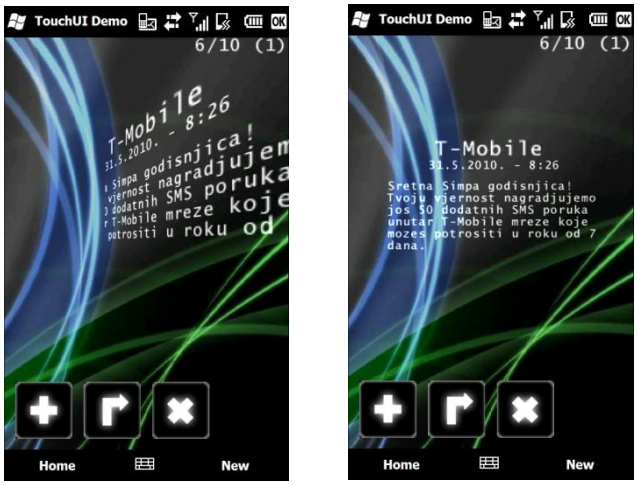
```
korak = 0
DOK JE (korak < N){
	kut = 90 / N * korak
	a = SIN(kut) * sirinaEkrana / 2
	b = COS(kut) * sirinaEkrana / 2
	StaviMatricuNaStog()
		TranslacijaPoX(-(sirinaEkrana / 2 - b))
		TranslacijaPoZ(-a)
		RotacijaOkoY(-kut)
		NacrtajTrenutniEkran()
	VratiMatricuSaStoga()
	StaviMatricuNaStog()
		TranslacijaPoX(sirinaEkrana - 2 * (sirinaEkrana / 2 - b))
		NacrtajSljedeciEkran()
	VratiMatricuSaStoga()
	korak++
}
```

Sljedeća u nizu animacija, pokreće se prilikom prelaska s ekrana prve razine na ekran druge razine i obrnuto. Pri tome se trenutni ekran približava kameri, počevši od trenutne pozicije i ujedno se povećava zbog perspektivne projekcije. U određenom trenutku nestaje i novo

odabrani ekran dolazi iz pozadine te zaustavlja se na početnoj poziciji trenutnog ekrana. Ova animacija uključuje samo translaciju i to po z-osi u pozitivnom smjeru. Pseudoalgoritam je vrlo jednostavan, a prikazani je u nastavku.

```
korak = 0
DOK JE (korak < N){
       AKO JE (korak <= N / 2){
              StaviMatricuNaStog()
                     TranslacijaPoZ(korak * indexZ)
                     NacrtajTrenutniEkran()
              VratiMatricuSaStoga()
       }
       INAČE {
              StaviMatricuNaStog()
                     TranslacijaPoZ(-(N * indexZ) + korak * indexZ)
                     NacrtajSljedeciEkran()
              VratiMatricuSaStoga()
       }
       korak++
}
```

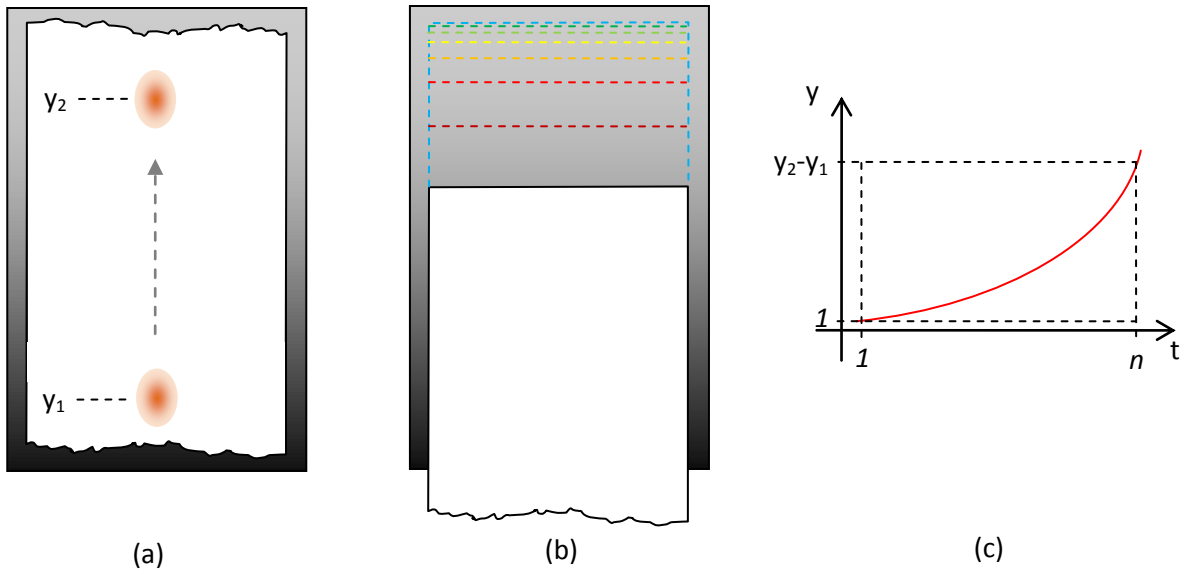
Osim samih ekrana, animirani su i pojedini njihovi elementi. Tako primjerice pritiskom na neku tipku na ekranu, tipka se uvećava što je rezultat translacije prema kameri za malu vrijednost. Zanimljiva animacija je i prilikom promjene SMS i Email poruke. Tekst poruke se najprije translatira prema kameri što rezultira povećanjem, nakon toga se poruka rotira oko y-osi za 180 stupnjeva te se na kraju translatira od kamere što rezultira smanjenjem teksta i vraćanjem na početnu poziciju. Na slici 3.8 prikazana je ova animacija kroz 2 slike.



Slika 3.8. Prikaz animacije kod promjene SMS poruke.

Ovo je primjer jedne vrlo zanimljive animacije koja zahtijeva samo rotaciju i translaciju bez bilo kakvih dodatnih izračuna. Posljednja u nizu animacija vezana je uz prikaz popisa programa i kontakata. S obzirom na broj programa, tj. kontakata, njihov cijeli popis nije

moguće prikazati na ekranu, već se prikazuju samo isječci. Da bi se osigurao dojam neprekidnog popisa potrebno je isti translatirati po y-osi ovisno o potezu prstom po ekranu. Vrijednost translacije računa se na temelju udaljenosti točke na kojoj je prst pritisnuti i točke na kojoj je prst podignuti što je i prikazano na slici 3.9. (a). Takav pomak prstom naziva se gesta. Više o gestama u [Touch Geastures, 2010].



Slika 3.9. Grafički opis translacije popisa po y-osi. (a) Određivanje pozicije pritiska i otpuštanja prsta. (b) Grafički prikaz postepenog zaustavljanja kretanja popisa. (c) Grafički prikaz eksponencijalne funkcije.

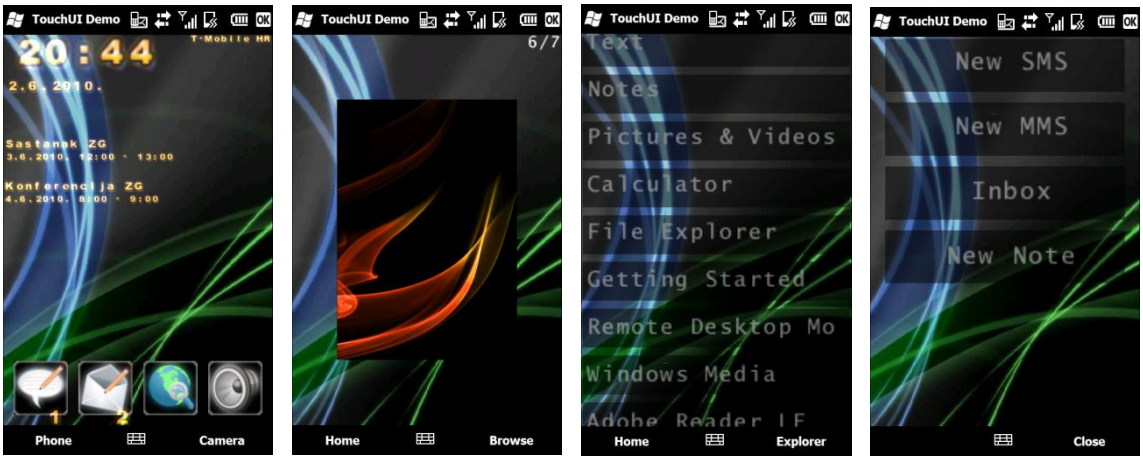
Vrijednost translacije jednaka je toj udaljenosti ukoliko je vrijeme između pritiska i otpuštanja prsta veće od zadanog (500ms). U protivnom se vrijednost translacije uvećava obrnuto proporcionalno vremenu, tj. što je kraće vrijeme između pritiska i otpuštanja prsta, to je duža translacija. Na taj način osiguravamo izvođenje translacije i nakon otpuštanja prsta. Da bismo izbjegli naglo zaustavljanje pomicanje popisa, potrebno je opet uključiti malo matematike. U ovom slučaju koristimo eksponencijalnu funkciju kojom simuliramo trenje, tj. popis se nakon otpuštanja prsta nastavlja kretati, ali se polagano zaustavlja (slika 3.9 (b)). Za ovu animaciju koristimo eksponencijalnu funkciju na sljedeći način. Na temelju vrijednosti ciljane translacije $(y_2 - y_1)$ i željenog broja koraka n izračunavamo bazu a eksponencijalne funkcije prema sljedećoj formuli.

$$(y_2 - y_1) = a^n$$
$$a = \sqrt[n]{(y_2 - y_1)}$$

Tako dobivenu bazu u svakom koraku animacije potenciramo korakom animacije i za dobivenu vrijednost translatiramo popis po y-osi. Pseudoalgoritam opisane animacije prikazani je u nastavku.

```
korak = 0
y2 = y2 + y2 * (maxTrajanjePomicanja - trajanjePomicanja) / 100
od = 1
do = y2 - y1
a = Korijen(do, N)
DOK JE (korak < N){
       StaviMatricuNaStog()
              y = y1 + Potenciraj(a, N - korak)
              TranslacijaPoY(y)
              Nacrtaj()
       VratiMatricuSaStoga()
       korak++
}
```

I za kraj nekoliko slika ekrana gotove aplikacije prikazano je na slici 3.10. Redom su prikazani početni ekran, ekran za pregled slika, ekran s popisom programa te ekran za tekstualne poruke.



Slika 3.10. Nekoliko ekrana gotove aplikacije

4. Rezultati

Rezultati ovog rada mogu se mjeriti na nekoliko načina. Prvi i najjednostavniji je mjerenje broja sličica koje se obrade u sekundi (engl. Frames per second). Što se veći broj sličica izmijeni u sekundi, aplikacija se izvodi brže i bez "trzanja". Druga mjera odnosi se na stabilnost sustava i može se mjeriti prema različitim parametrima. Primjerice broj grešaka u jedinici vremena, tip pogreške, zauzeće procesora i memorije i sl. Treća mjera je subjektivnog karaktera i odnosi si se na zadovoljstvo korisnika u radu s aplikacijom. Za potrebe ovog rada testiranje je izvršeno na mobilnom uređaju HTC Touch Pro 2 s rezolucijom ekrana od 480x800 piksela, procesorom od 528MHz, 288MB radne memorije, od čega 64MB otpada na video memoriju, te Windows Mobile 6.5 operacijskim sustavom.

4.1. FPS

Izmjerene FPS vrijednosti ovise o dva ključna faktora. Prvi Faktor se odnosi na procesorsku i memorijsku snagu uređaja. Što uređaj ima više procesorske snage i više memorije, veći je i rezultat za FPS. Drugi faktor odnosi se na kvalitetu programskog koda, tj. ovisi o stupnju optimiziranosti koda. To znači da inteligentnim programiranjem, vodeći računa o svakoj varijabli te zauzeću procesora i memorije možemo uvelike uštedjeti na resursima potrebnim za izvođenje aplikacije. Važno je naglasiti da se FPS vrijednost konstantno mijenja i ovisi prvenstveno o objektima koji se trenutno iscrtavaju na zaslonu, njihovoj veličini, broju i kompleksnosti. U tablici 4.1 možemo vidjeti u kojoj mjeri pojedina animacija utječe na performanse izvođenja aplikacije. Izmjereni rezultati su očekivani i u potpunosti se poklapaju sa složenošću pojedine animacije i resursima potrebnim za njezino izvođenje. Vrijednosti u drugoj koloni pokazuju za koliko se smanji FPS vrijednost prilikom izvođenja animacije. Nakon završetka animacije, vrijednost se vraća na početnu, koja pak ovisi o složenosti ekrana koji se prikazuje. FPS vrijednosti za pojedini ekran prikazane su u tablici 4.2.

Tablica 4.1. Utjecaj animacije na smanjenje FPS vrijednosti

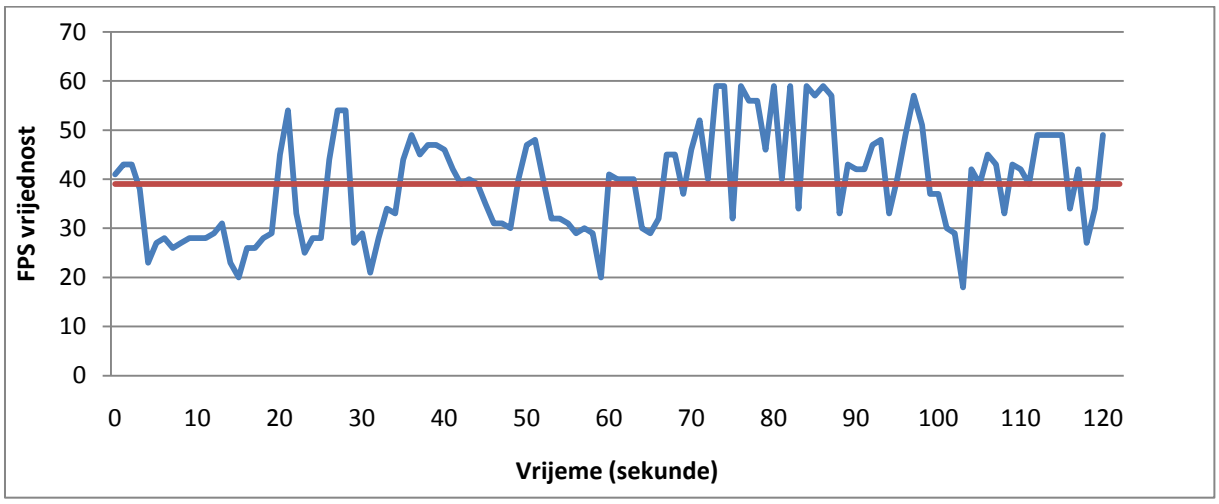
Animacija	Smanjenje FPS (prosječno)
Level 1 animation	-7
Level 2 animation	-4
Message scroll animation	-2
Contacts list scroll animation	-2

I ovi podaci su također u skladu s očekivanjima, jer su u direktnoj vezi sa brojem elemenata koji su prikazani na ekranu. Što se više elemenata treba iscrtati na ekran, posebno teksta, to je FPS vrijednost manja.

Tablica 4.2. FPS vrijednosti u pojedinom trenutku izvođenja aplikacije

Prikazani ekran/animacija	Razina ekrana	FPS (prosječno)
Home screen	1	43
Programs screen	1	30
Contact screen	1	27
SMS screen	1	34
Email screen	1	32
Music screen	1	52
Pictures screen	1	60
Text messaging	2	49
Email messaging	2	58
Internet	2	46
Multimedia	2	49
Contact details	2	53

Minimalno izmjerena vrijednost je 27 i ukoliko od toga oduzmemo još 7 za animaciju prve razine, dobivamo minimalnu vrijednost od 20 FPS. S obzirom da je donja granica oko 30 FPS, valjalo bi provesti analizu programskog koda i pokušati smanjiti složenost. Iz navedenih rezultata jasno je da FPS vrijednost varira prilikom izvođenja aplikacije. Na slici 4.1. prikazani je graf izmjerenih FPS vrijednosti tijekom dvominutnog korištenja aplikacijom.



Slika 4.1. Izmjerene FPS vrijednosti prilikom korištenja aplikacije u trajanju od dvije minute

4.2. Stabilnost sustava

Druga mjera odnosi se na stabilnost sustava. Provedenom analizom izvođenja aplikacije utvrđeno je da se zauzeće procesore kreće u rasponu od 50-60%, što znači da je potrebno provesti analizu programskog koda kako bi se utvrdilo zbog čega je postotak toliko visok. Ukoliko postoji mogućnost da se vrijednost smanji, poboljšanja bi se zasigurno primijetila i kod mjerenja FPS vrijednosti.

4.3. Zadovoljstvo korisnika

Posljednja u nizu predloženih mjera je zadovoljstvo korisnika. Nakon što završi faza izrade korisničkog sučelja, korisnost i upotrebljivost istog potrebno je provjeriti u suradnji s korisnicima. Korisnici imaju priliku isprobati rad s novim sučeljem kako bi nam mogli dati povratnu informaciju o njihovom doživljaju i mogućim nepravilnostima koje je potrebno ispraviti. Osim samog vizualnog identiteta, sučelje treba osigurati i ispravno, odnosno za korisnika funkcionalno rješenje. Jednom riječju, sučelje treba biti upotrebljivo. Postoji mnogo tehnika za procjenu zadovoljstva korisnika [Stone, 2005.], no najkorištenije su zasigurno neformalni razgovor s korisnikom, promatranje korisnika prilikom korištenja aplikacije, intervjuiranje korisnika te popunjavanje upitnika [Jones, 2006., str. 197-206]. Za potrebe ovog rada odabrana je prva i najlakša metoda, a to je brzi i neformalni razgovor s korisnikom pri čemu mu je pokazana aplikacija, a od njega se očekivalo da ukratko prokomentira prvi dojam. Iako je ova metoda vrlo jednostavna i temeljena na malom uzorku korisnika, može poslužiti kao prvi test kvalitete razvijene aplikacije. Podaci koji su prikupljeni za potrebe ovog rada pokazuju da su korisnici zadovoljni brzinom i funkcionalnošću pokazane aplikacije, dok za grafički, tj. vizualni dio očekuju poboljšanja u sljedećoj verziji.

5. Budući rad

Budući rad u velikoj je mjeri vezan uz probleme koji su otkriveni prilikom mjerenja rezultata. To se prije svega odnosi na visoku potrošnju procesorske snage i niske FPS vrijednosti. Isto tako, nije provedena odgovarajuća evaluacija od strane korisnika kojom bi se prikupili vrlo važni podaci vezani uz zadovoljstvo korisnika te njihove prijedloge za poboljšanja.

Osim otklanjanja problema, u svrhu poboljšanja aplikacije, potrebno je dodati nove funkcionalnosti koje bi dodatno olakšale korištenje mobilnim uređajem. Isto tako, prilagodbom za izvođenje na drugim platformama, proširuje se i krug potencijalnih korisnika aplikacije, a time i kvaliteta i kvantiteta povratne informacije.

Još je jednom važno naglasiti kako je svrha ovog rada pokazati značenje matematike u računalnoj grafici i naglasak je stavljen na vizualnom identitetu aplikacije, dok je njezina funkcionalnost stavljena u drugi plan. Stoga nema svrhe uspoređivati ovu aplikaciju s aplikacijama slične namjene koje su komercijalno dostupne i razvijene od strane velikih korporacija, u čijem razvoju sudjeluje mnoštvo ljudi.

6. Zaključak

Suvremeni mobilni uređaji privlače korisnike velikim ekranom osjetljivim na dodir iza kojeg se krije grafički bogato korisničko sučelje. Za izradu takvog sučelja nije dovoljno samo dizajnersko znanje, veći i dobro poznavanje matematike koja čini srž 2D i 3D prikaza te animacija. Suvremene grafičke biblioteke, kao što su OpenGL ili DirectX, uvelike olakšavaju cijeli postupak razvoja, jer sakrivaju od korisnika komplicirane matrične izračune. Takve grafičke biblioteke uvelike olakšavaju i samu pripremu zaslona i koordinatnog sustava za prikaz elemenata, a korisniku, tj. programeru, je ostavljena mogućnost oblikovanja tih modela i njihov smještaj u prostor. Bez obzira na to, za postizanje dinamike na sceni, potrebno je dobro poznavanje matematike, posebice geometrije kako bi se na ekranu uređaja mogle kvalitetno simulirati pojave iz stvarnosti.

Ovaj rad daje naglasak upravo na taj dio računalne grafike pokazujući kako se korištenjem jednostavnih matematičkih izraza, poput sinusa, kosinusa, eksponencijalne funkcije i sl. mogu dobiti zanimljivi efekti na grafičkom korisničkom sučelju, dajući mobilnom uređaju sasvim novu dimenziju.

Popis literature

1. Astle, D., Durnil, D. (2004.), OpenGL ES Game Development, Course Technology PTR
2. Ballard, B. (2007), *Designing the Mobile User Experience*, Wiley
3. Ducker, M. (2000.), Matrix and Vector Manipulation for Computer Graphics. Dostupno 30.5. na http://www.gamedev.net/reference/articles/download/course1.pdf
4. Fling, B. (2009), *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps*, O'Reilly Media.
5. Fosner, R. (1996.), OpenGL Without the Pain: Creating a Resuable 3D View Class for MFC. Dostupno 30.5. na http://www.microsoft.com/msj/archive/S2085.aspx
6. International Telecomunication Union, *Global ICT developments*, dostupno 30.5.2010. na http://www.itu.int/ITU-D/ict/statistics/ict/
7. Jones, M., Marsden, G. (2006.), *Mobile Interaction Design*, Wiley
8. Joubert, N., Andrews, J. (2008.), Viewing and Camera Control in OpenGL. Dostupno 30.5.2010. na http://njoubert.com/teaching/cs184_fa08/section/sec09_camera.pdf
9. Leech, J. (2010.), Khronos Native Platform Graphics Interface, Dostupno 30.5.2010. na http://www.khronos.org/registry/egl/specs/eglspec.1.4.20100405.pdf
10. Malizia, A. (2006.), Mobile 3D Graphics, Springer
11. Munchi, A., Leech, J. (2008.), OpenGL ES Common/Common-Lite Profile Specification Version 1.1.12 (Full Specification), Dostupno 30.5.2010. na http://www.khronos.org/registry/gles/specs/1.1/es_full_spec_1.1.12.pdf
12. Pulli, K., Aarnio T., Miettinen, V., Roimela, K., Vaarala, J. (2007.), *Mobile 3D graphics with OpenGL ES and M3G*, Morgan Kaufman
13. Segal, M., Akeley, K. (2004.), The OpenGL Graphics System: A Specification. Dostupno 30.5.2010. na http://www.opengl.org/documentation/specs/version2.0/glspec20.pdf
14. Stone D., Jarret C., Woodroffe M., Minocha S. (2005.), User Interface Design and Evaluation, Morgan Kaufmann Publishers
15. Touch Geastures (2010.). Dostupno 30.5.2010. na http://msdn.microsoft.com/en-us/library/ee207148.aspx

Sažetak

Autor:

Marko Škvorc

Naslov rada:

Razvoj naprednih grafičkih korisničkih sučelja na mobilnim uređajima

Tekst sažetka:

Osnovna ideja ovog rada je razvoj grafički bogatog korisničkog sučelja na mobilnom uređaju korištenjem suvremenih grafičkih biblioteka s naglaskom na primjeni matematičkih znanja u svrhu izvođenja grafičkih transformacija na zaslonu mobilnog uređaja.

Ključne riječi:

mobilni uređaj, grafičko korisničko sučelje, OpenGL ES, trigonometrija

Summary

Author:

Marko Škvorc

Title:

Developing rich graphical user interface on mobile devices

Abstract:

The basic idea is to develop a rich graphical user interface on mobile device using modern graphical libraries and mathematical knowledge for different graphical transformations on a mobile screen.

Key words:

mobile device, graphical user interface, OpenGL ES, trigonometry

Životopis

Marko Škvorc

Rođen 29. svibnja 1986. godine u Čakovcu. Nakon završetka prirodoslovno matematičke gimnazije u Čakovcu, 2005. godine upisao Fakultet organizacije i informatike u Varaždinu. Tri godine kasnije završava preddiplomski studij s titulom prvostupnik informatike i kao jedan od 10% najboljih studenata na godini dobiva pohvalu dekana. Iste godine upisuje diplomski studij na istom Fakultetu, smjer Informacijsko i programsko inženjerstvo. 2009. godine se kao član tima natječe na svjetskom finalu Imagine Cup natjecanja u Kairu, Egipat, gdje osvaja Windows Mobile Award za projekt GeoScout.Net. Na temelju tog uspjeha dobiva posebno priznanje rektora Sveučilišta i dekana Fakulteta. Paralelno sa studijem se i certificira kao MCTS (Microsoft Certified Technology Specialist) za Microsoft .NET Framework 3.5.