SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1178

KOMBINACIJA STVARNIH I VIRTUALNIH SCENA

Eva Cetinić

Zagreb, lipanj 2010.

Sadržaj

1. Uvod 3

2. Razine stvarnosti 4

3. Proširena stvarnost – definicija i karakteristike 5

3.1. Miješanje slike 5

3.1.1. Optičko miješanje 6

3.1.2. Video miješanje 6

3.1.3. Proširena stvarnost na zaslonu 7

3.1.4. Projekcijska proširena stvarnost 8

3.2. Poravnavanje u 3D 8

4. Metode praćenja objekata 9

4.1. Metode praćenja računalnim vidom i obradom snimaka 9

4.1.1. Metode praćenja s markerima 9

4.1.1.1. Ravninski pravokutni marker 10

4.2. Metode praćenja bez markera 11

5. Programski alati za izradu proširene stvarnosti 12

5.1. ARToolKit 12

5.1.1. Načela razvoja aplikacije 13

5.2. NyARToolkit 14

5.3. FLARToolkit 14

6. Primjeri kombinacija virtualnih i stvarnih scena 15

6.1. Osnovni dijelovi programa 16

6.2. Varijacija virtualnih objekata 24

6.3. Transformacija objekata u sceni 28

6.4. Višestruki markeri 33

7. Primjene proširene stvarnosti 38

8. Zaključak 43

9. Literatura 44

10. Sažetak 46

1. Uvod

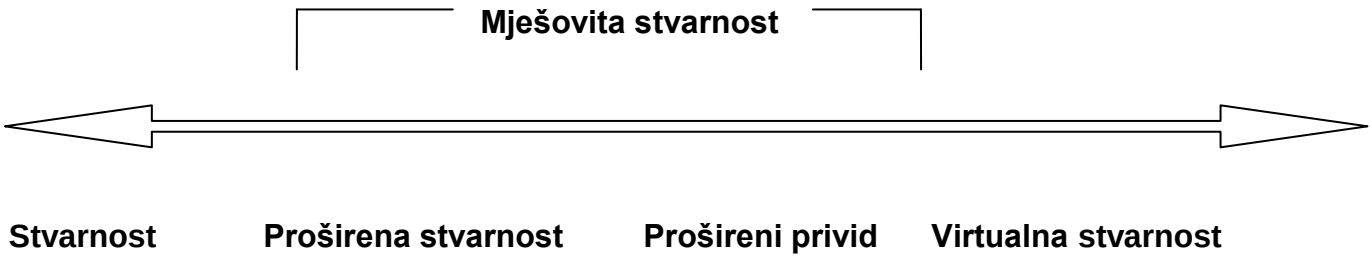
U današnje vrijeme pojam stvarnosti postaje sve manje egzaktan. Tehnološki razvoj pridonosi sve stvarnijoj realizaciji nestvarnoga te stvarnost više nije samo jedna, ona se proteže kroz paletu više ili manje „stvarnih stvarnosti“. Unaprjeđenjem tehnologija računalne grafike svijet virtualne stvarnosti postaje sve realističniji, a mogućnost njegovog kombiniranja sa stvarnim okruženjem našeg svijeta otvara novu dimenziju u razumijevanju koncepta stvarnosti – mješovitu stvarnost. Mješovita stvarnost ostvaruje se kao kombinacija virtualnih i stvarnih scena. Ovu kombinaciju moguće je ostvariti implementacijom elemenata stvarnog okruženja u virtualne scene (prošireni privid) ili implementacijom virtualnih objekata u stvarno okruženje (proširena stvarnost). U ovom radu bit će naglasak na teorijskom objašnjenju i praktičnom ostvarenju pojma proširene stvarnosti.

2. Razine stvarnosti

Razine stvarnosti najbolje se opisuju konceptom virtualnog kontinuuma stvarnosti (Milgramov kontinuum[1]). Virtualni kontinuum stvarnosti opisuje postojanje kontinuirane skale u rasponu od potpuno virtualnog do potpuno stvarnog okruženja, odnosno prikaz praktične klasifikacije raznih srodnih tehnika.

U tom rasponu razlikujemo nekoliko razina, odnosno pojmova:

- Virtualna stvarnost (Virtual Reality - VR)
  - korisnik vidi virtualno okruženje
  - koriste se samo slike dobivene računalnom grafikom i animacijom
- Prošireni privid (Augmentet Virtuality – AV)
  - korisnik vidi virtualno okruženje s elementima stvarnog
  - snimljeni dijelovi stvarnog svijeta uključeni su u prividni svijet
- Proširena stvarnost (Augmented Reality – AR)
  - korisnik vidi stvarno okruženje s elementima virtualnog
  - računalna grafika nanesena na slike stvarnog svijeta
- Mješovita stvarnost (Mixed Reality – MR)
  - proširena stvarnost i prošireni privid
- Stvarnost (Reality)
  - korisnik vidi prikaz stvarnog okruženja



[1] Miligramov kontinuum stvarnog i virtualnog definirali su 1994. godine Paul Miligram i Fumio Kishino

3. Proširena stvarnost – definicija i karakteristike

Proširena stvarnost temelji se na dodavanju elemenata virtualnog okruženja u stvarni svijet tako da oni djeluju kao dio stvarnog svijeta. To je način da se korisnikovo viđenje stvarnog svijeta proširi dodatnim informacijama – računalno generiranim multimedijskim sadržajem (tekst, slika, grafika).

Osnovne karakteristike proširene stvarnosti su:

- Kombinacija stvarnog i virtualnog
- Interakcija u stvarnom vremenu
- Poravnavanje u 3D

Ove karakteristike nužno trebaju biti ispunjene da bi se nešto moglo smatrati produktom proširene stvarnosti. Npr., razne tehnike koje se koriste u filmskoj industriji kojima se u stvarne scene snimljene kamerom naknadno dodaju virtualni elementi, iako slične tehnikama koje se koriste u proširenoj stvarnosti, ne spadaju u domenu proširene stvarnosti upravo zbog uvjeta interakcije u stvarnom vremenu.

Za valjanu realizaciju proširene stvarnosti potrebno je riješiti tri osnovna problema: miješanje slike (dobivanje istovremenog prikaza virtualnih i stvarnih scena), poravnavanje te prikupljanje podataka.

3.1. Miješanje slike

Miješanje slike moguće je ostvariti na više načina od kojih su najvažniji:

- Optičko miješanje
- Video miješanje
- Proširena stvarnost na zaslonu
- Projekcijska proširena stvarnost

3.1.1. Optičko miješanje

U ostvarenju ovog načina istovremenog prikaza virtualne i stvarne scene potrebno je da korisnik na glavi nosi uređaj za prikaz. Uređaj za prikaz sadrži optičku miješalicu, odnosno poluprozirno ogledalo takovo da je moguće istovremeno vidjeti dvije slike. Izravno kroz ogledalo korisnik vidi sliku stvarnog svijeta, a u ogledalu se odražava slika virtualne scene sa zaslona montiranog na odgovarajući način. Za svako oko postoji jedan zaslon čime se ostvaruje stereo slika. Slika virtualne scene dolazi s grafičkog protočnog sustava koje se nalazi na računalu. Uređaj za prikaz kablom je spojen sa računalom što znači da korisniku nije omogućen visok stupanj pokretljivosti. Na uređaju za prikaz montiran je i slijednik položaja i orijentacije glave koji potrebne podatke šalje računalu koje na temelju toga može generirati sliku iz odgovarajuće perspektive.

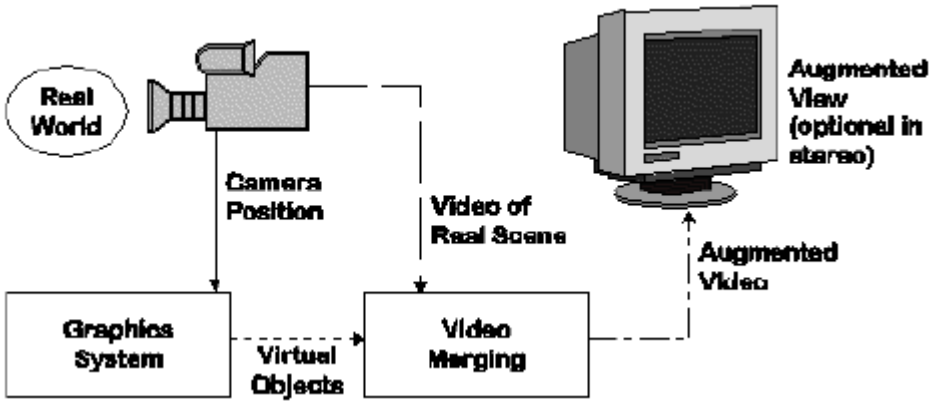
3.1.2. Video miješanje

Kod video miješanja, slično kao kod optičkog miješanja, korisnik na glavi nosi uređaj za prikaz (Slika 1). U ovom slučaju korisnik vidi samo sliku koja dolazi sa zaslona. Slika stvarnog svijeta dolazi iz para kamera montiranih na uređaju, a slika virtualnog svijeta generira se jednako kao kod optičkog miješanja. Virtualna i stvarna slika kombiniraju se u video miješalici gdje se obje nalaze u digitalnom obliku što omogućuje slobodniju obradu signala i postizanje raznih efekata. Stvarna se slika obradom i analizom može upotrijebiti za određivanje položaja korisnika. Nedostatak ovog mehanizma je da u slučaju kvara ili isključenja sustava korisnik gubi sliku stvarnog svijeta, takav nagli gubitak vida može biti sigurnosni problem.

Slika 1. HMD (Head Mounted Display)

3.1.3. Proširena stvarnost na zaslonu

Ovo je najjednostavnija izvedba proširene stvarnosti. Princip je isti kao kod video miješanja, međutim pogled nije upravljan pokretima glave nego pokretima kamere. Slika najčešće nije stereoskopska, odnosno postoji samo jedna slika stvarnog svijeta i jedna virtualna slika (Slika 2). Upravo zbog svoje jednostavnosti i praktičnosti ovakva je izvedba proširene stvarnosti našla najveći broj komercijalnih primjena i širok spektar programskih alata za izradu.



Slika 2. Princip rada proširene stvarnosti na zaslonu

Izvedenica proširene stvarnosti na zaslonu je proširena stvarnost na ručnom zaslonu. Kamera je postavljena na ručni uređaj koji korisniku prikazuje scenu stvarnog svijeta proširenu dodatnim informacijama, odnosno virtualnim elementima.

3.1.4. Projekcijska proširena stvarnost

Kod projekcijske proširene stvarnosti korisnici najčešće ne nose dodatnu opremu na glavi već su dodatne informacije, tj. virtualne scene, prikazane izravno u stvarnom okruženju korištenjem jednog ili više projektora. Najčešće se za projekciju koriste ravne plohe u radnom okruženju. Tehnike analize slike omogućuju praćenje pokreta korisnika i interakciju. Ako se virtualni elementi projektiraju na neravne površine potrebno je prethodno sliku deformirati na odgovarajući način kako bi se kompenzirala deformacija nastala projekcijom.

3.2. Poravnavanje u 3D

Poravnavanje je najteži problem proširene stvarnosti. Radi se o tome da je potrebno precizno poravnati stvarne i virtualne predmete, i to ne na zaslonu nego u 3D prostoru. Virtualna scena konstruira se s koordinatnim sustavom koji točno odgovara koordinatnom sustavu stvarnog svijeta. Tako se položaji stvarnih predmeta izravno prenose u virtualnu scenu i virtualni predmeti se postavljaju u odnosu na njih. Također, položaj kamere (oko promatrača) u stvarnom svijetu prenosi se istim koordinatama u virtualnu scenu i upotrebljava kao položaj virtualne kamere. Bez potpunog poznavanja 3D strukture stvarne okoline nije moguće postići ispravno preklapanje predmeta po dubini što predstavlja suštinsku razliku između 2D poravnavanja u završnoj slici i poravnavanja u 3D sceni. Prilikom poravnavanja potrebna je izrazita preciznost, naime ljudsko oko detektira pomake manje od jedne kutne minute stoga je će i najmanja nepreciznost biti primijećena. Na primjer, ako želimo virtualni predmet postaviti na stvarni stol mala nepreciznost u slijeđenju

položaja korisnika može rezultirati neusklađenošću stvarne i virtualne scene, odnosno moguće je da predmet lebdi iznad stola ili potone u njega.

Osnova procesa poravnavanja jest praćenje, odnosno je postupak dobivanja položaja i orijentacije premeta u stvarnom vremenu.

4. Metode praćenja objekata

Glavne tehnike praćenja objekata su:

- računalni vid i obrada snimaka
- mehaničko praćenje
- magnetsko praćenje
- ultrazvučno praćenje
- inercijsko praćenje

4.1. Metode praćenja računalnim vidom i obradom snimaka

Razlikujemo dva načina praćenja objekata: praćenje s markerima i praćenje bez markera.

4.1.1. Metode praćenja s markerima

Princip metoda praćenja s markerima temelji se na tome da se na ciljne objekte ili dijelove scene postave oznake, npr. LED diode ili reflektirajuće vrpce. Zadaća markera jest dobivanje informacije iz slike i procjena položaja praćenog objekta. Postavljanje markera kao metoda označavanja mjesta za dodavanje virtualnih objekata prikladnija je u zatvorenim prostorima nego otvorenim.

Vrste markera:

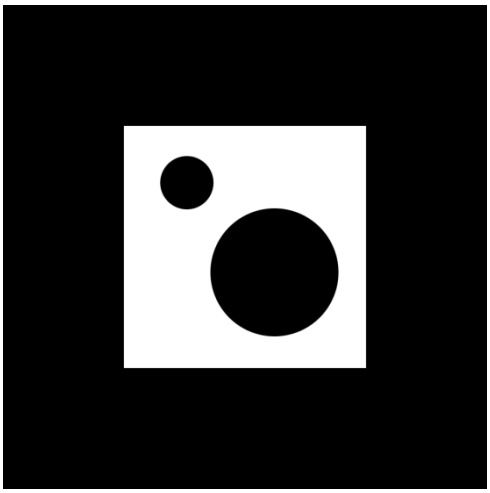
- Točkasti
    - pridruživanje scene i slike preko markera
    - kružni oblik
    - nepromjenjivost oblika kod perspektivnih izobličenja
    - lakše prepoznavanje raznih uređenih geometrijskih uzoraka

- Ravninski
    - pretvorba područja ograničenog s označenim vrhovima u ciljnu površinu

4.1.1.1. Ravninski pravokutni marker

Metoda praćenja ravninskog pravokutnog markera osnovni je princip funkcioniranja programskih alata za ostvarenje proširene stvarnosti opisanih u ovom radu. Ravninski pravokutni marker nazivan još i „magičnim simbolom“[2], jeftino i na smetnje otporno rješenje za praćenje objekata u prostoru u stvarnom vremenu. Sastoji se od crnog ruba na bijeloj podlozi koji olakšava otkrivanje objekata. Unutar crnog ruba nalazi se uzorak specifičan za pojedini marker. Princip rada je sljedeći: dobivena slika scene se binarizira, zatim se utvrđuju povezana područja crnih piksela, zadržavaju se područja slike čiji vanjski obris pristaje uz četiri linijska segmenta te se vrši popravak pojedinog područja da se izbjegnu perspektivna izobličenja. Nakon toga slijedi usporedba dobivenog uzorka iz slike sa poznatim uzorkom markera. Postupak otkrivanja markera do procjene položaja izvodi se brzinom od 24 slike u sekundi.

[2] *Magic symbol* tehnologiju, koja predstavlja okosnicu razvoja tehnologija proširene stvarnosti, razvila je tvrtka Inition 2006. godine



Slika 3. Primjer markera

4.2. Metode praćenja bez markera

Markeri su relativno neprikladni za postavljanje u otvorenoj i nepoznatoj okolino, tada se koriste metode kod kojih se praćenje zasniva na prirodnim značajkama ciljnih objekata kao što su rubovi, vrhovi ili teksture. Problem koji se javlja kod korištenja ovakvih metoda jesu poteškoće u otkrivanju prirodnih značajki objekata, npr. ponekad nije dostupan dovoljan broj točaka ili rubova na objektu, kamera se kreće prebrzo što uzrokuje zamućenje slike, jakost rasvjete se mijenja između slika, odbijanje i lomovi svjetla dovode do zabuna u tumačenju i sl. Znanja o ciljnom objektu pomažu u njegovom otkrivanju, npr. ako se radi o poznatom 2D ili 3D modelu objekta.

5. Programski alati za izradu proširene stvarnosti

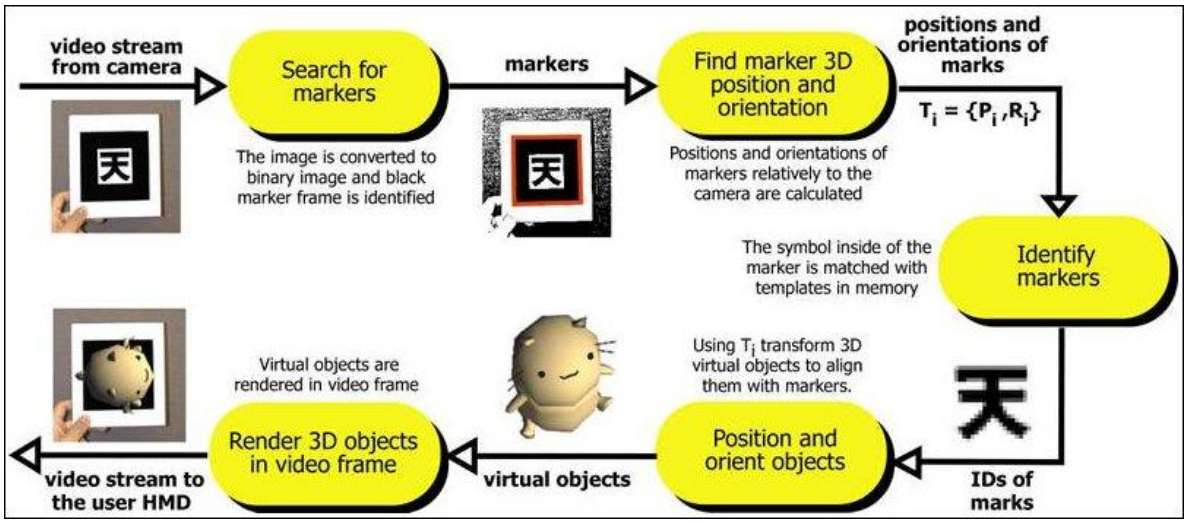
U ovom poglavlju bit će opisani najpoznatiji alati za realizaciju proširene stvarnosti. Svi ovi alati temelje se na principu raspoznavanja ravninskih pravokutnih markera.

5.1. ARToolKit

ARToolKit je biblioteka funkcija namijenjena za korištenje s programskim jezicima C i C++ koja omogućuje razvoj aplikacija proširene stvarnosti. Dostupan je za korištenje na svim platformama. ARToolkit razvio je dr. Hirokazu Kato s Tehnološkog laboratorija za tehnologije ljudskih sučelja na Sveučilištu u Washingtonu (SAD) 1999. godine, no postao je dostupan na korištenje široj javnosti tek 2003. Razlog tomu bila je nestabilnost i specifična primjena biblioteke funkcija, koja se koristila kao dio istraživačkog okvira namijenjenog razvijanju izvedivog oblika proširene stvarnosti.

ARToolKit aplikacije omogućuju preklapanje virtualnih prikaza i realnog svijeta. U tome ključnu ulogu igraju markeri. Proces praćenja se odvija na sljedeći način:

1. Kamerom se zabilježi video stvarnog svijeta i šalje na računalo.
2. Programska podrška na računalu pretražuje svaki blok videa tražeći četverokutni oblik crne boje.
3. Ako je četverokut pronađen, programska podrška korištenjem određenog algoritma izračunava relativnu poziciju kamere u odnosu na četverokut.
4. Kada je poznata pozicija kamere iscrtava se grafički model (virtualni objekt) s te poziciju kamere.
5. Model se iscrtava povrh videa stvarnog svijeta te tako djeluje kao da se nalazi na markeru.
6. Konačni izlaz prikazuje se na zaslonu, korisnik kada gleda na zaslonu vidi kombinaciju virtualne i stvarne scene.



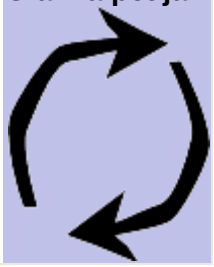
Slika 4. Princip rada ARToolKit-a

5.1.1. Načela razvoja aplikacije

Razvoj aplikacija koje koriste ARToolKit sastoji se od dva dijela: pisanje aplikacije i osposobljavanje programa obrade slika s markerima koji se koriste u aplikacijama.

Pisanje aplikacija s ARToolKitom temelji se na jednostavnom obrascu. Koraci koji su nužno prisutni u osnovnom kodu izrade aplikacije su:

Tablica 1. - Koraci algoritma

Inicijalizacija	1. Inicijaliziraj video snimku (započni snimanje) i učitaj datoteku s uzorkom markera i parametre kamere.
Glavna petlja 	2. Dohvati ulazni video blok.
	3. Detektiraj marker i raspoznaj uzorak u ulaznom video bloku.
	4. Izračunaj transformacije kamere u odnosu na prepoznati uzorak
	5. Iscrtaj virtualni objekt na detektiranom uzorku.
Kraj	6. Završi snimanje.

5.2. NyARToolkit

NyARToolKit je inačica ARToolkit-a napisana isključivo programskim jezikom Java. Zbog toga se izvodi sporije od originala, no može se koristiti na svim dostupnim platformama. Baš kao i originalni alat, NyARToolKit je biblioteka funkcija za vizualnu interpretaciju i integraciju virtualnih podataka u stvarne fizičke okoline, uključujući funkcionalnost korištenja kamere u stvarnom vremenu, 3D iscrtavanje virtualnih objekata i integriranje u izlazni tok podataka. Ovu biblioteku funkcija razvio je 2008. godine japanski programer Nyatla, otuda potječe naziv alata, odnosno dodatak prefiksa Ny. Komercijalna licenca NyARToolKita u vlasništvu je kompanije ARToolworks, Inc, koja je komercijalizirala i ARToolKit, iako u niti jednom obliku ne sudjeluje u razvijanju istog. Vjerojatno se upravo zbog toga NyARToolKit distribuira kao nekomercijalni *open source* proizvod.

5.3. FLARToolkit

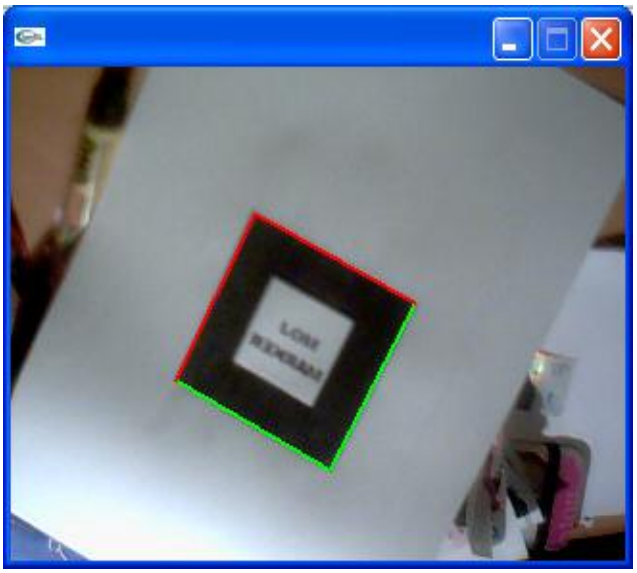
Flash-based Augumented Reality Toolkit, FLARToolKit je alat proširene stvarnosti temeljen na Flash tehnologiji. Razvili su ga japanski programeri u drugoj polovici 2008. godine. Iako je slobodno dostupan javnosti, još uvijek za njega ne postoji nikakva dokumentacija. Alat je kompatibilan sa verzijom 10 Flasha i koristi Action script 3. Međutim, kao i sa svim Flash programima, izvodi se iznimno sporo, pogotovo u usporedbi sa verzijama napisanima za programske jezike C i Java. Kao i alati iz kojih je razvijen, FLARToolKit koristi Magic Symbol tehnologiju za pozicioniranje elemenata proširene stvarnosti u snimkama stvarne scene.

6. Primjeri kombinacija virtualnih i stvarnih scena

Za izradu ovdje opisanih primjera potrebno je okruženje za rad sa C programskim jezikom (korišten je MS Visual Studio 2005), instalirana web kamera, glut-3.7.6. biblioteka, te ARToolKit alat.

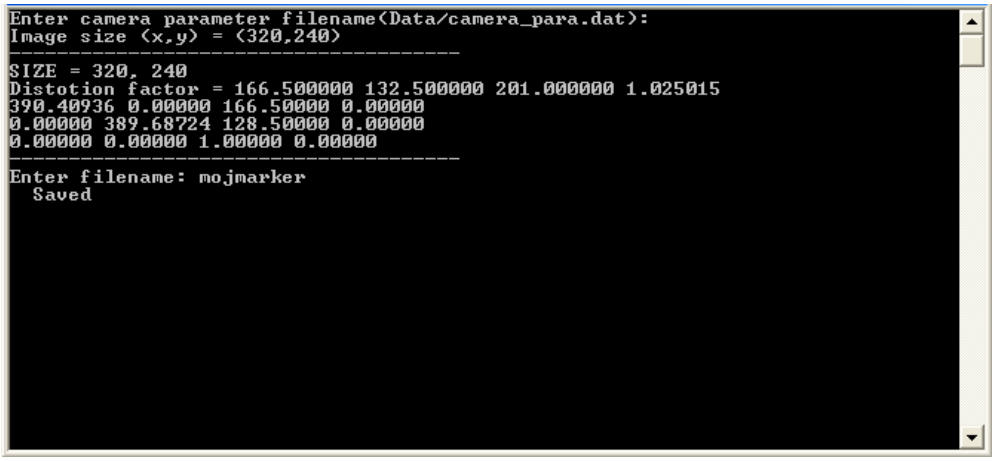
ARToolKit alat moguće je preuzeti sa http://www.hitl.washington.edu/artoolkit. U preuzetom direktoriju nalaze se sve potrebne datoteke koje omogućuju funkcioniranje programa te također skup već gotovih predložaka markera. Moguće je snimiti i vlastite markere. Za snimanje markera koristi se datoteka *mk_patt.exe***.** Nakon njenog pokretanja na ekranu se pojavljuje direktna slika s web kamere. Nakon što program pronađe marker on ga označava sa zeleno-crvenim okvirom, te se pritiskom lijeve tipke miša sprema marker. Potrebno je da cijeli marker bude vidljiv na ekranu te da papir s markerom ne bude savinut.

Slika 5. Primjer vlastito izrađenog markera



Slika 6. Slika s web kamere u trenutku prepoznavanja markera

Nakon detekcije markera kamerom slijedi njegovo spremanje.



Slika 7. Spremanje vlastito snimljenog markera

Nakon unosa naziva taj se marker automatski sprema u direktorij ARToolKita kao datoteka *mojmarker* i može se koristiti u budućim programima kao jedan od markera koje alat može prepoznati.

6.1. Osnovni dijelovi programa

Osnovni dijelovi programa bit će objašnjeni na jednostavnom programu, *simple.c,* čijim pokretanjem se aktiviran web kamera, na zaslonu se vidi snimka stvarne okoline, a na mjestu gdje se nalazi marker dodaje se virtualna kocka.

Programske funkcije koje odgovaraju koracima algoritma opisanima u *Tablici 1* opisane su sljedećom tablicom:

.

Tablica 2 – Programske funkcije koje odgovaraju koracima algoritma ARToolKit aplikacija

ARToolKit korak	**Funkcija**
1. Inicijalizacija aplikacije.	`init`
2. Dohvat video bloka.	`arVideoGetImage` (poziva se u `mainLoop`)
3. Detektiranje markera.	`arDetectMarker` (poziva se u `mainLoop`)
4. Izračun transformacija kamere.	`arGetTransMat` (poziva se u `mainLoop`)
5. Iscrtavanje virtualnog objekta	`draw` (poziva se u `mainLoop`)
6. Završetak hvatanje video zapisa.	`cleanup`

Najvažnije funkcije u programu su `main`, `init`, `mainLoop`, `draw` i `cleanup` koje će biti detaljnije objašnjene:

- `main`

Glavna funkcija programa *simple.c* :

```
main(int argc, char *argv[])
{
    init();
    arVideoCapStart();
    argMainLoop( NULL, keyEvent, mainLoop );
}
```

Funkcija `init` sadrži dio koda za pokretanje hvatanja video zapisa, učitavanje markera i parametara kamere, te određivanje postavki grafičkog prozora. To odgovara koraku inicijalizacije. Nadalje se u ulazi u stanje “stvarnog vremena”, tj. pozivom funkcije `arVideoCapStart` započinje snimanje fizičke okoline u stvarnom vremenu. Zatim se pozivom `argMainLoop` funkcije pokreće glavna programska petlja i povezuje funkcija `keyEvent` sa aktivnostima na tipkovnici te petlja `mainLoop` sa glavnom petljom grafičkog iscrtavanja.

- Init

Init funkcija poziva se iz main funkcije i služi za inicijalizaciju hvatanja video zapisa i učitavanje inicijalnih ARToolKit parametara.
Isprva se otvara video put i pronalazi veličina video slike:

```
/* otvori video put */
if( arVideoOpen( vconf ) < 0 ) exit(0);

/* pronađi veličinu prozora */
if( arVideoInqSize(&xsize, &ysize) < 0 ) exit(0);
printf("Image size (x,y) = (%d,%d)\n", xsize, ysize);
```

Varijabla `vconf` sadrži inicijalnu video konfiguraciju i definirana je na vrhu programa *simple.c*.
Nakon toga potrebno je inicijalizirati parametre ARToolKit aplikacije.
Ključni parametri su:

- uzorci koji će biti korišteni na obrascu uzoraka, tj. markeru, i virtualni objekti koji će odgovarati tim uzorcima
- karakteristike kamere koja se koristi

Oni se mogu učitati iz imena datoteka koje su specificirane u komandnoj liniji ili korištenjem standardno kodiranih datoteka.

Parametri kamere učitavaju se preko standardno definirane datoteke `Data/camera_para.dat`:

```
/* postavi inicijalne parametre kamere */
if( arParamLoad(cparaname, 1, &wparam) < 0 ) {
    printf("Camera parameter load error !!\n");
    exit(0);
}
```

Zatim se ti parametri transformiraju s obzirom na trenutnu veličinu slike jer se parametri kamere mijenjaju ovisno o veličine slike.

```
arParamChangeSize( &wparam, xsize, ysize, &cparam );
```

Parametri kamere se postavljaju na učitane vrijednosti i ispisuju na ekran:

```
arInitCparam( &cparam );
printf("*** Camera Parameter ***\n");
arParamDisp( &cparam );
```

Nakon toga slijedi učitavanje određenog uzorka markera. U ovom slučaju koristimo datoteku *mojmarker* dobivenu snimanjem vlastitog markera pomoću *mk_patt.exe*

```
if( (patt_id=arLoadPatt(patt_name)) < 0 ) {
    printf("pattern load error !!\n");
    exit(0);
}
```

S time da je na početku programa obavljena inicijalizacija varijabli:

```
char  *patt_name      = "Data/mojmarker";
int    patt_id;
```

Naposljetku se otvara grafički prozor:

```
/* otvori graficki prozor */
argInit( &cparam, 1.0, 0, 0, 0, 0 );
```

Drugi argument funkcije `argInit` definira zoom funkciju, ako je postavljen na 1.0 onda odgovara formatu video slike, ako je postavljen na 2.0 onda udvostručava veličinu okvira.

- `mainLoop`

Glavna petlja je dio u kojem se odvija glavnina poziva ARToolKit funkcija i sadrži dio koda koji odgovara koracima od 2 do 5 iz *Tablice 1.*

Prvo se dohvaća video okvir korištenjem funkcije `arVideoGetImage`:

```
/* dohvati video okvir  */
if( (dataPtr = (ARUint8 *)arVideoGetImage()) == NULL ) {
    arUtilSleep(2);
    return;
}
```

Video slika se zatim prikazuje na zaslonu. Ona može biti neiskrivljena ili iskrivljena slika koja ispravlja smetnje kamere. Iskrivljavanje slike proizvodi kvalitetniju sliku, ali može značajno utjecati na brzinu izmjene video okvira. U ovom primjeru slika nije iskrivljena:

```
argDrawMode2D();
argDispImage( dataPtr, 0,0 );
```

Zatim se koristi funkcija `arDetectMarker` za pretraživanje video slike u potrazi za markerom s odgovarajućim uzorkom.

```
if( arDetectMarker(dataPtr, thresh, &marker_info, &marker_num) < 0 )
{
    cleanup();
    exit(0);
});
```

Broj pronađenih markera sadržan je u varijabli `marker_num`, dok je `marker_info` pokazivač na listu struktura markera koja sadrži informacije o koordinatama, vrijednostima pouzdanosti prepoznavanja i identifikacijskim brojevima za svaki marker.

U ovom trenutku izvođenja programa video slika se prikazuje i analizira stoga ju nije više potrebno koristiti. Za dohvat sljedeće slike, tj. video okvira koristi se funkcija:

```
arVideoCapNext();
```

Nadalje, sve vrijednosti pouzdanosti detektiranih markera se uspoređuju kako bi se ispravan identifikacijski broj markera povezao sa najvišom vrijednosti pouzdanosti. Provjerava se vidljivost markera s odgovarajućim uzorkom jer o vidljivosti markera ovisi i vidljivost virtualnog objekta. Ako se u stvarnoj sceni snimanoj kamerom marker ne pojavljuje u potpunosti, tj. ako nije vidljiv čitavi crni okvir markera, objekt se neće iscrtati.

```
  /* provjeri vidljivost markera */
  k = -1; // vrijednost kad nije pronaden odgovarajuci uzorak markera

for( j = 0; j < marker_num; j++ ) {
    if( patt_id == marker_info[j].id ) {
        if( k == -1 ) k = j;
        else if( marker_info[k].cf < marker_info[j].cf ) k = j;
    }
}

if( k == -1 ) {
    argSwapBuffers();
    return;
}
```

Za dobivanje transformacija između oznake markera u sceni i kamere koristi se funkcija `arGetTransMat`:

```
/* nadi transformacije između markera i kamere */
arGetTransMat(&marker_info[k], patt_center, patt_width, patt_trans);
```

Stvarna pozicija kamere i relativni pomak u odnosu na marker sadržani su u 3x4 matrici `patt_trans`.

Naposljetku se virtualni objekt iscrtava na oznaku markera korištenjem draw funkcije kojoj se kao parametar prenose izračunate transformacije:

```
draw(patt_trans);
```

Napomena: jednostavan optimizacijski korak omogućuje da se preskoči poziv `draw` funkcije i direktno izvrši izmjena spremnika ako u sceni nije pronađen odgovarajući uzorak markera (k== -1):

```
if( k == -1 ) {
    argSwapBuffers();
    return; }
```

- draw

Funkcija iscrtavanja podijeljena je u tri segmenta: inicijalizacija iscrtavanja, definiranje matrice i iscrtavanje objekta.

Inicijalizacija 3D iscrtavanja vrši se pozivima ARToolKit funkcija za iscrtavanje 3D objekata i korištenjem OpenGL funkcija za određivanje osnovnih postavki spremnika:

```
argDrawMode3D();
argDraw3dCamera( 0, 0 );
glClearDepth( 1.0 );
glClear(GL_DEPTH_BUFFER_BIT);
glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
```

Nakon toga potrebno je konvertirati izračunatu transformacijsku matricu (3x4 matrica) u OpenGL format (niz od 16 brojeva) korištenjem funkcije `argConvGlpara`. Tih 16 brojeva predstavlja pozicijske i orijentacijske vrijednosti realne kamere i koriste se za određivanje pozicije virtualne kamere kako bi se izjednačio položaj realne i virtualne kamere te tako postiglo da se virtualna i realna scena podudaraju i da se virtualni objekt iscrtava točno na poziciji realnog markera.

```
/* ucitaj transformacijsku matricu */
argConvGlpara(patt_trans, gl_para);
glMatrixMode(GL_MODELVIEW);
glLoadMatrixd( gl_para );
```

Pozicija virtualne kamere postavlja se korištenjem OpenGL funkcije `glLoadMatrixd(gl_para)`.

U zadnjem dijelu koda izvršava se iscrtavanje 3D objekta, u ovom primjeru to je kocka plave boje na bijelom svjetlu.

```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glLightfv(GL_LIGHT0, GL_AMBIENT, ambi);
glLightfv(GL_LIGHT0, GL_DIFFUSE, lightZeroColor);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash);
glMaterialfv(GL_FRONT, GL_SHININESS, mat_flash_shiny);
glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
glMatrixMode(GL_MODELVIEW);
glTranslatef( 0.0, 0.0, 25.0 );
glutSolidCube(50.0);
```

Vrijednosti varijabli `light_position, ambi, lightZeroColor, mat_flash, mat_flash_shiny, mat_ambient` definirane su na početku funkcije `draw()`.

Za kraj je potrebno resetirati prethodno definirane OpenGL varijable:

```
glDisable( GL_LIGHTING );
glDisable( GL_DEPTH_TEST );
```

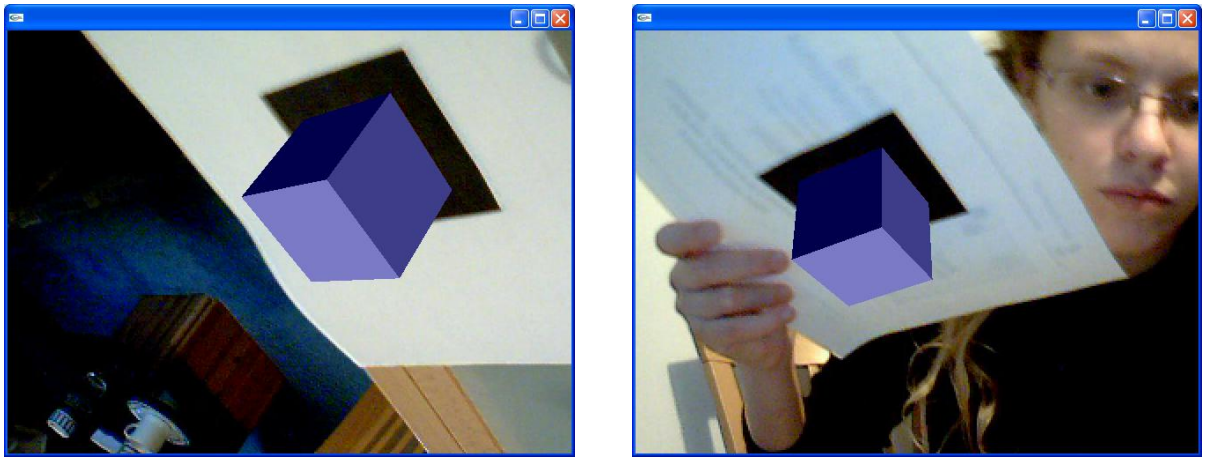
- `cleanup`

Ova funkcija se poziva kod zaustavljanja video procesa i zatvaranja video puta kako bi postao slobodan pristup drugim aplikacijama.

```
arVideoCapStop();
arVideoClose();
argCleanup();
```

Rezultat izvođenja programa:

(ovdje je prikazan *screen shot* ekrana u trenutku hvatanja video signala sa web kamere)



Slika 8. Slike na ekranu dobivene s kamere u trenutku izvođenja programa

6.2. Varijacija virtualnih objekata

Prethodno opisan primjer *simple.c* moguće je modificirati tako da virtualni objekt više nije kocka plave boje, već bilo koje drugo trodimenzionalno geometrijsko tijelo ili objekt iz biblioteke *glut.h.* To se postiže tako da se u funkciji draw umjesto naredbe `glutSolidCube();` koristi neka od slijedećih funkcija:

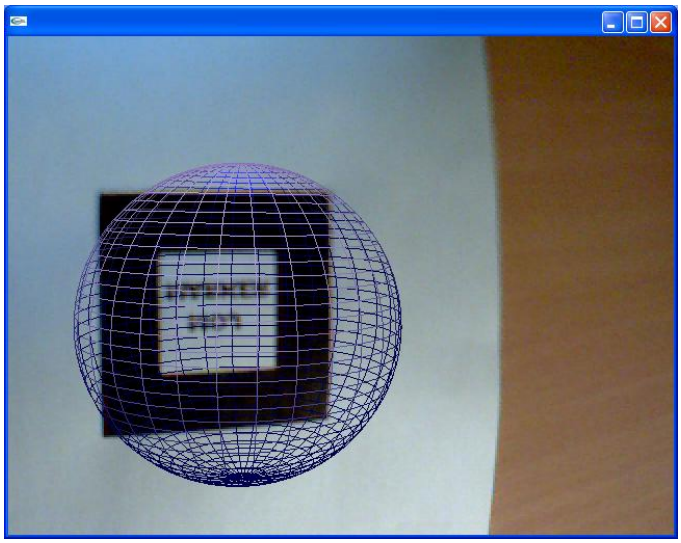
```
/* GLUT pre-built models sub-API */
GLUTAPI void APIENTRY glutWireSphere(GLdouble radius, GLint slices,
GLint stacks);
GLUTAPI void APIENTRY glutSolidSphere(GLdouble radius, GLint slices,
GLint stacks);
GLUTAPI void APIENTRY glutWireCone(GLdouble base, GLdouble height,
GLint slices, GLint stacks);
GLUTAPI void APIENTRY glutSolidCone(GLdouble base, GLdouble height,
GLint slices, GLint stacks);
GLUTAPI void APIENTRY glutWireCube(GLdouble size);
GLUTAPI void APIENTRY glutSolidCube(GLdouble size);
GLUTAPI void APIENTRY glutWireTorus(GLdouble innerRadius, GLdouble
outerRadius, GLint sides, GLint rings);
GLUTAPI void APIENTRY glutSolidTorus(GLdouble innerRadius, GLdouble
outerRadius, GLint sides, GLint rings);
GLUTAPI void APIENTRY glutWireDodecahedron(void);
GLUTAPI void APIENTRY glutSolidDodecahedron(void);
GLUTAPI void APIENTRY glutWireTeapot(GLdouble size);
GLUTAPI void APIENTRY glutSolidTeapot(GLdouble size);
GLUTAPI void APIENTRY glutWireOctahedron(void);
GLUTAPI void APIENTRY glutSolidOctahedron(void);
GLUTAPI void APIENTRY glutWireTetrahedron(void);
GLUTAPI void APIENTRY glutSolidTetrahedron(void);
GLUTAPI void APIENTRY glutWireIcosahedron(void);
GLUTAPI void APIENTRY glutSolidIcosahedron(void);
```

Također je moguće mijenjati postavke svjetla scene i boju iscrtavanja objekta promjenom parametara varijabli definiranih na početku funkcije draw():

```
GLfloat   mat_ambient[]     = {0.0, 0.0, 1.0, 1.0};
GLfloat   mat_flash[]       = {0.0, 0.0, 1.0, 1.0};
GLfloat   mat_flash_shiny[] = {50.0};
GLfloat   light_position[]  = {100.0,-200.0,200.0,0.0};
GLfloat   ambi[]            = {0.1, 0.1, 0.1, 0.1};
GLfloat   lightZeroColor[]  = {0.9, 0.2, 0.9, 0.1};
```

Npr. rezultat izvođenja programa pri korištenju naredbe

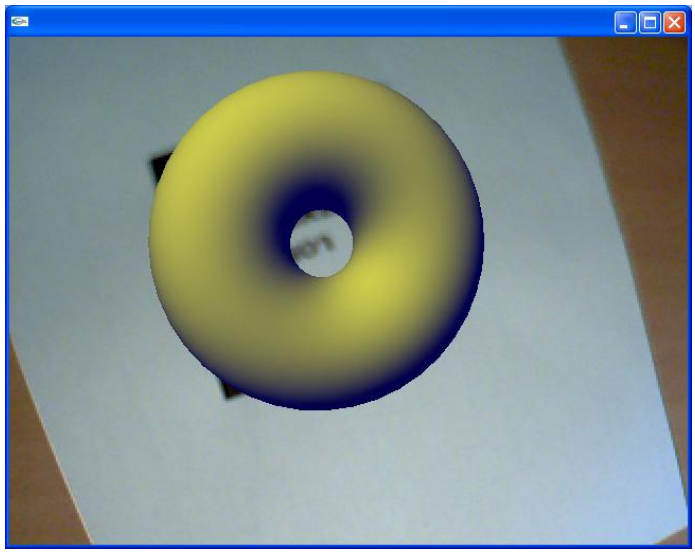
`glutWireSphere(50.0, 30.0, 40.0):`



Slika 9. WireSphere

Rezultat izvođenja programa pri korištenju naredbe

`glutSolidTorus(20.0, 30.0, 40.0, 40.0):`

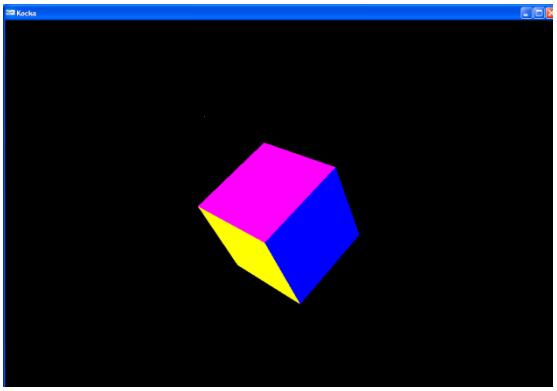


Slika 10. SolidTorus

Rezultat izvođenja programa pri korištenju naredbe `glutSolidTeapot(50.0);`:

Slika 11. Virtualni čajnik

Također moguće je umjesto gotovih glut funkcija kreirati vlastite modele korištenjem OpenGL naredbi za iscrtavanje mnogokuta. U sljedećem primjeru implementiran je 3D model kocke sa stranicama različite boje sastavljene od šest četverokuta. Rezultat programa koji iscrtava kocku:

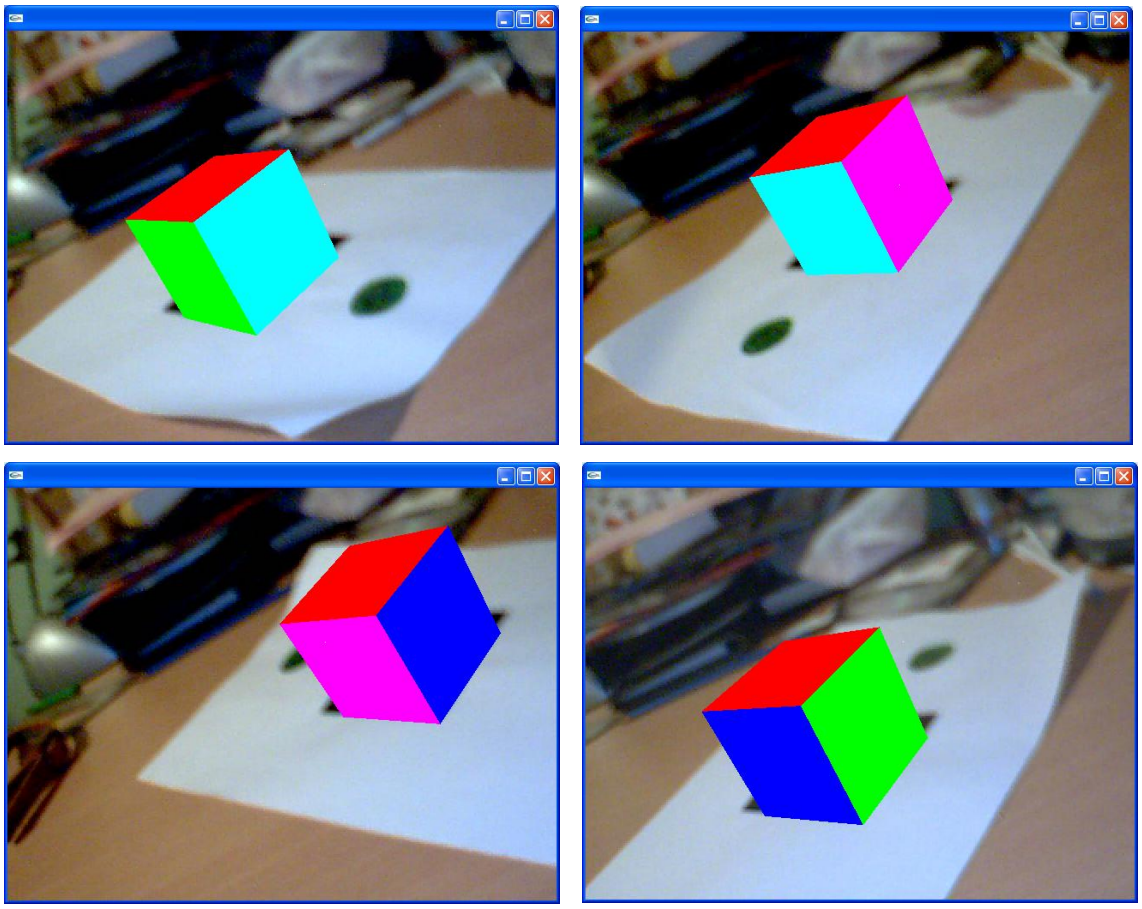


Slika 12. Kocka sastavljena od kvadrata različite boje

Ako se u draw funkciji na mjestu prethodno navedenih gotovih funkcija za iscrtavanje napiše:

```
glBegin(GL_QUADS); //zapocni unos koordinata cetverokuta

//koordinate se odreduju u odnosu na srediste kvadrata

glColor3f(1.0f,0.0f,0.0f);//crvena
glVertex3f(0.5f, 0.5f, 0.5f);
glVertex3f(0.5f, -0.5f, 0.5f);
glVertex3f(-0.5f, -0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, 0.5f);

glColor3f(0.0f,0.0f,1.0f);//plava
glVertex3f(0.5f, 0.5f, 0.5f);
glVertex3f(0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, 0.5f, 0.5f);

glColor3f(1.0f,0.0f,1.0f);//ruzicasta
glVertex3f(0.5f, 0.5f, 0.5f);
glVertex3f(0.5f, 0.5f, -0.5f);
glVertex3f(0.5f, -0.5f, -0.5f);
glVertex3f(0.5f, -0.5f, 0.5f);

glColor3f(0.0f,1.0f,0.0f);//zelena
glVertex3f(-0.5f, 0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, -0.5f, -0.5f);
glVertex3f(-0.5f, -0.5f, 0.5f);

glColor3f(1.0f,1.0f,0.0f);//zuta
glVertex3f(0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, -0.5f, -0.5f);
glVertex3f(0.5f, -0.5f, -0.5f);

glColor3f(0.0f,1.0f,1.0f);//tirkiz
glVertex3f(0.5f, -0.5f, 0.5f);
glVertex3f(0.5f, -0.5f, -0.5f);
glVertex3f(-0.5f, -0.5f, -0.5f);
glVertex3f(-0.5f, -0.5f, 0.5f);


glEnd(); //zavrsi unos koordinata cetverokuta
```

U stvarnoj sceni će se na mjestu markera iscrtati šarena kocka. Zanimljivo je primijetiti kako se okretanjem papira na kojem se nalazi marker, odnosno pomicanjem markera okreće i kocka (za lakše uočavanje pomicanja s jedne strane papira naznačen je znak kruga). Slike prikazuju video isječke iz snimke tijekom koje se papir markerom pomicao u smjeru kazaljke na satu.



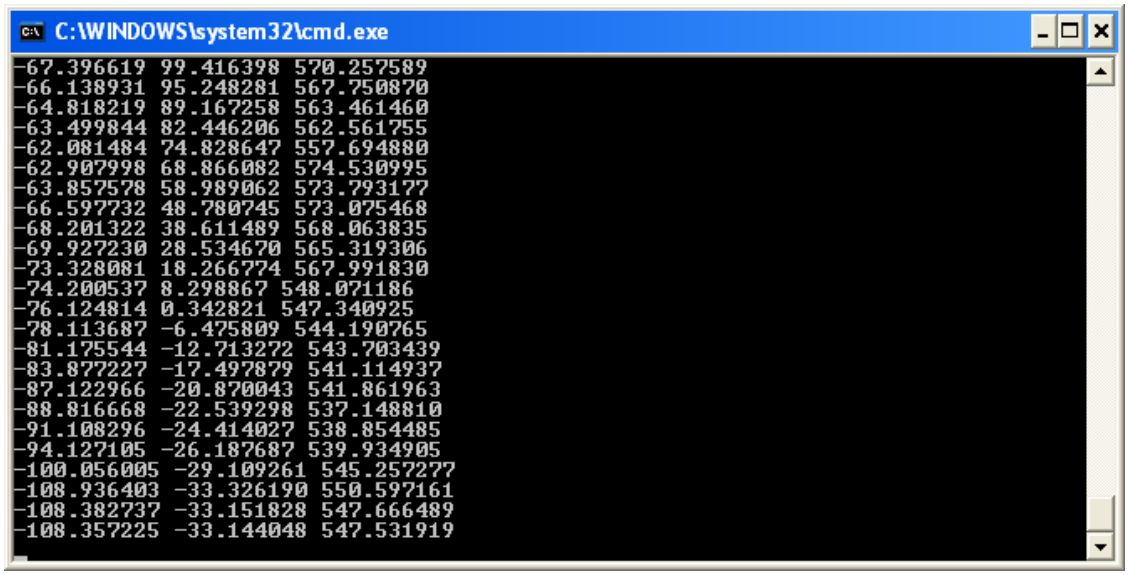
Slika 13. Okretanje virtualne raznobojne kocke u stvarnoj sceni

6.3. Transformacija objekata u sceni

ARToolKit određuje poziciju markera u koordinatnom sustavu kamere i koristi OpenGL sustav matrica za određivanje pozicije virtualnog objekta. Tijekom izvođenje programa u glavnoj petlji konstantno se izračunava matrica transformacija koja određuje odnos položaja markera i položaja kamere. Za izračunavanje matrice koristi se funkcija `arGetTransMat`. Ako u kodu nakon poziva te funkcije pozovemo funkciju:

```
printf("%f %f %f\n",patt_trans[0][3],patt_trans[1][3],patt_trans[2][3]);
```

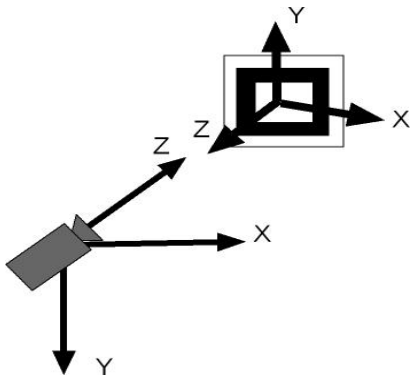
u komandnom prozoru tijekom izvođenja programa ispisivat će se vrijednosti transformacijske matrice na poziciji trećeg stupca i prvog, drugog i trećeg retka.



Slika 14. Vrijednosti tranformacijske matrice

Ako pomičemo marker u odnosu na kameru prema lijevo povećava se prva od tri ispisane vrijednosti, ako ga pomičemo prema gore povećeva se druga vrijednost, a ako ga pomičemo prema naprijed povećava se zadnja vrijednost, odnosno faktor u zadnjem retku matrice. Ta se matrica prenosi kao argument funkciji `draw`, funkciji u čijim se pozivom izvršava iscrtavanje virtualnog objekta, gdje se onda pretvara u OpenGL format matrica.

Koordinatni sustavi markera i kamere opisani su sljedećom slikom:



Slika 15. Koordinatni sustavi kamere i markera

Koordinatni sustav markera ima istu orijentaciju kao i OpenGL koordinatni sustav stoga sve transformacije koje se primjenjuju na objekt moraju zadovoljavati principe OpenGL transfromacija. Koordinatni sustav virtualne kamere, odnosno onaj koordinatni sustav u odnosu na koji se iscrtava željeni objekt, postavljen je tako da se njegovo ishodište nalazi u središtu markera. Kada želimo da se virtualni objekt nalazi točno na markeru, tj. da izgleda kao da je pričvršćen za marker, potrebno je prije iscrtavanja objekta izvršiti translaciju koordinatnog sustava.

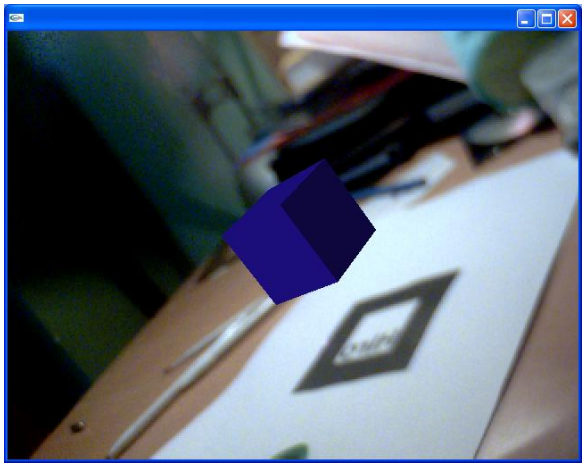
```
glMatrixMode(GL_MODELVIEW);
glTranslatef( 0.0, 0.0, 25.0 );  // translacija po osi z
glutSolidCube(50.0);
```

Budući da virtualni objekt kocke ima dimenzije stranica 50.0 potrebno ju je translatirati za 25.0 po z osi kako bi se prilikom izvođenja programa na markeru iscrtala cijela kocka koja svojom donjom stranicom dodiruje površinu markera, tj. da izgleda kao da se cijela kocka nalazi na markeru.

Moguće je translatirati objekte za bilo koji iznos, npr.:

```
glMatrixMode(GL_MODELVIEW);
glTranslatef( 0.0, 0.0, 75.0 );  // translacija po osi z
glutSolidCube(50.0);
```

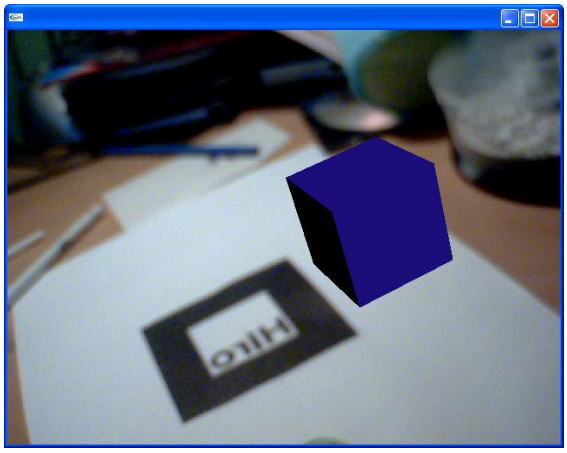
Ako translatiramo kocku za 75.0 po z osi ona će izgledati kao da lebdi iznad markera:



Slika 16. Translacija kocke po z osi

Također možemo ju translatirati u smjeru bilo koje osi, npr. u smjeru x osi:

```
glMatrixMode(GL_MODELVIEW);
glTranslatef( 75.0, 0.0, 25.0 );  // translacija po osi x i z
glutSolidCube(50.0);
```



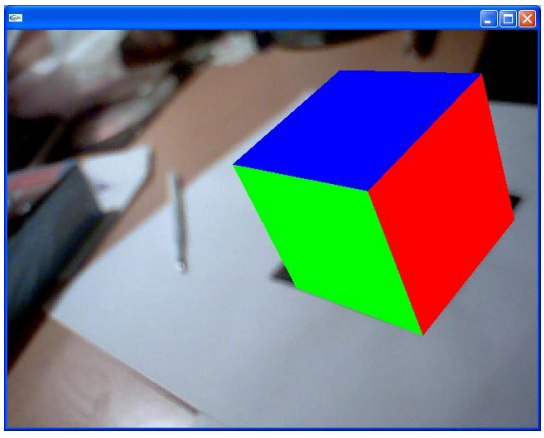
Slika 17. Translacija kocke po x i z osi

Osim translacije moguće je raditi i ostale transformacije, odnosno rotaciju i skaliranje objekta. Npr. ako želimo okrenuti šarenu kocku iz prethodnog primjera možemo ju rotirati oko x osi naredbom:

```
glRotatef (angle, 1.0, 0.0, 0.0);
```

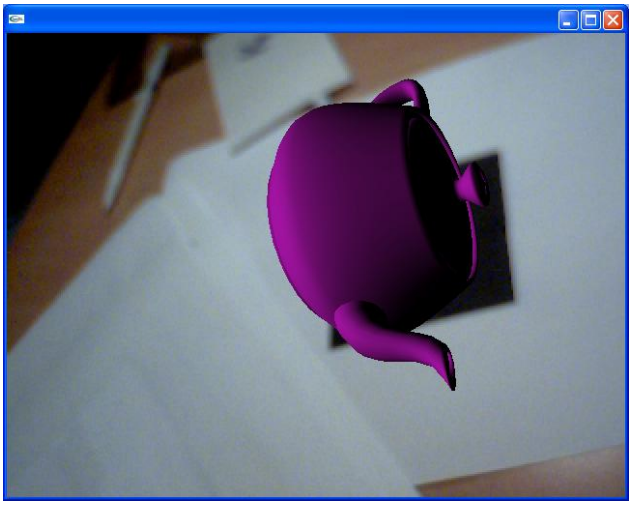
gdje je `angle` varijabla definirana kao kut od 90 stupnjeva.

Kocka je sada okrenuta tako da se gore nalazi stranica plave boje:



Slika 18. Rotacija kocke za 90 stupnjeva oko x osi

Transformacije su naročito bitne kod modela koji nemaju pravilni geometrijski oblik. Transfromacijama možemo postići da se virtualni modeli raznolikih oblika što prikladnije uklope u stvarnu scenu. Npr. da prije iscrtavanja modela čajnika nisu izvršene potrebne transformacije (rotacija za 90 stupnjeva oko x osi) rezultat izvođenja programa bio bi sljedeći:

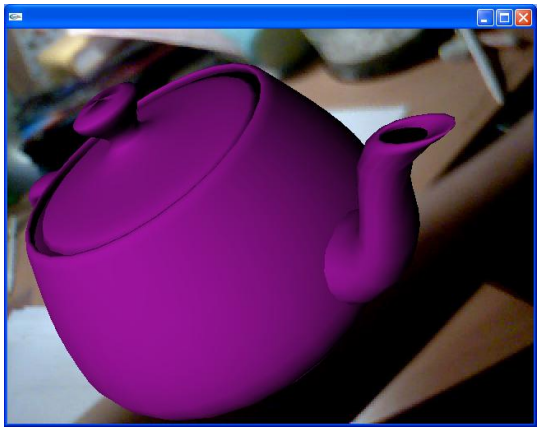


Slika 19. Zaokrenuti čajnik

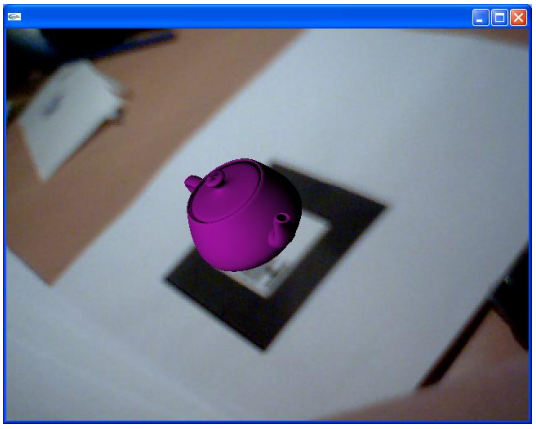
Zanimljivo je promotriti rezultate dobivene skaliranjem objekata. Npr. čajnik možemo povećati tako da izdužimo osi koordinatnog sustava tri puta ili ga umanjiti tako da dva puta smanjimo dužinu osi:

```
glTranslatef( 0.0, 0.0, 25.0 );
glScalef (3.0, 3.0, 3.0);
glRotatef (angle, 1.0, 0.0, 0.0);
glutSolidTeapot(50.0);
```

```
glTranslatef( 0.0, 0.0, 25.0 );
glScalef (0.5, 0.5, 0.5);
glRotatef (angle, 1.0, 0.0, 0.0);
glutSolidTeapot(50.0);
```



Slika 20. Uvećani čajnik



Slika 21. Umanjeni čajnik

6.4. Višestruki markeri

Ako želimo u stvarnoj sceni imati više markera povezanih s različitim virtualnim 3D objektima potrebno je izvršiti određene modifikacije prvotnog kôda *simple.c* kao što su:

- učitavanje datoteke s deklariranim višestrukim uzorcima
- nova struktura povezana s uzorcima koja implicira drugačiji kôd provjere i transformacije u programu
- redefinicija sintakse i definicije `draw` funkcije

Za učitavanje više uzoraka u ARToolKitu postoji definirana specifična funkcija `read_ObjData` sadržana u datoteci *object.c* koju se uključuje u glavni program. Zatim se iz glavnog programa učitavanje markera obavlja pozivom:

```
if( (object=read_ObjData(model_name, &objectnum)) == NULL ) exit(0);
printf("Objectfile num = %d\n", objectnum);
```

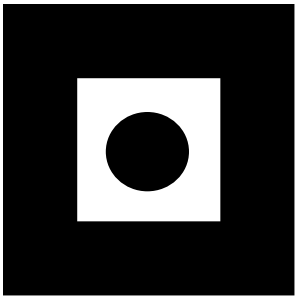
Na početku glavnog programa nalaze se definicije pojedinih varijabli. `Object` je pokazivač na `ObjectData_T` strukturu. To je specifična struktura kreirana za listu uzoraka. Na početku programa varijabla `model_name` definirana je preko datoteke *object_data2.*koja sadrži informacije o markerima.

```
char            *model_name = "Data/object_data2";
ObjectData_T    *object;
int             objectnum;
```

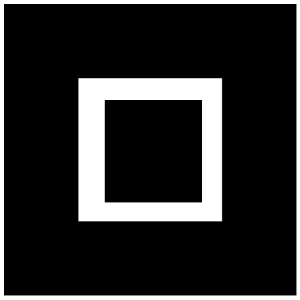
U tekstualnoj datoteci *object_data2* koja se koristi u ovom primjeri opisane su opisane su specifičnosti markera koji će se koristiti u sceni, u ovom slučaju koriste se dva različita markera. Za svaki pojedini marker navedeni su:

- ime
- naziv datoteke u kojem je spremljen
- širina markera
- koordinate centra uzorka markera

U ovom primjeru korištena su dva vlastito izrađena markera, spremljena datotekama *sfera* i *cube.*



Slika 22. Marker sfera



Slika 23. Marker cube

Kada se u stvarnoj sceni pojavi marker *sfera* na njemu će se iscrtati kugla, kada se pojavi marker *cube* na njemu će se iscrtati kocka.

Sadržaj datoteke *object_data2*:

```
#the number of patterns to be recognized
2

#pattern 1
sphere
Data/sfera
80.0
0.0 0.0

#pattern 2
cube
Data/cube
80.0
0.0 0.0
```

Linije datoteke koje započinju sa # su komentari i ignoriraju se prilikom učitavanja.

ARToolKit sada može identificirati navedene markere. Potrebno je provjeriti vidljivost svakog od navedenih markera i odrediti transformacije kamere u odnosu na oba markera. Ovisno o vidljivosti pojedinog markera zastavica za vidljivost njemu pripadajućeg objekta postavlja se na 0 ako nije vidljiv, odnosno 1 ako jest vidljiv.

Dio kôda glavne petlje u kojem se vrše navedene operacije:

```
/* provjeri vidljivost markera */
 for( i = 0; i < objectnum; i++ ) {
    k = -1;
    for( j = 0; j < marker_num; j++ ) {
        if( object[i].id == marker_info[j].id ) {
            if( k == -1 ) k = j;
            else if( marker_info[k].cf < marker_info[j].cf ) k = j;
        }
    }
     if( k == -1 ) {
        object[i].visible = 0;
        continue;
    }

    object[i].visible = 1;
    arGetTransMat(&marker_info[k],
    object[i].marker_center, object[i].marker_width,
    object[i].trans);
}
```

Nakon što je poznata vidljivost pojedinog markera i transformacija kamere u odnosu na marker poziva se funkcija draw kojoj se kao parametri šalju `ObjectData_T` struktura `object` i broj objekata.

```
draw( object, objectnum );
```

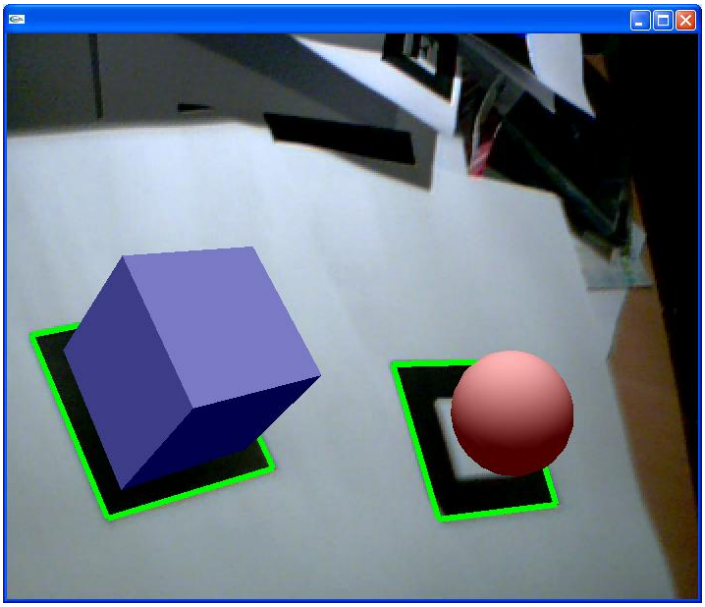
Funkcija `draw` definirana je na sljedeći način:

```
static int draw( ObjectData_T *object, int objectnum )
{
    int     i;
    double  gl_para[16];

      glClearDepth( 1.0 );
      glClear(GL_DEPTH_BUFFER_BIT);
      glEnable(GL_DEPTH_TEST);
      glDepthFunc(GL_LEQUAL);
      glEnable(GL_LIGHTING);

    /* izracunaj parametre transformacije */
    for( i = 0; i < objectnum; i++ ) {
        if( object[i].visible == 0 ) continue;
        argConvGlpara(object[i].trans, gl_para); // transformacija
        ARToolKit formata matrice u OpenGL format
        draw_object( object[i].id, gl_para);
    }
      glDisable( GL_LIGHTING );
      glDisable( GL_DEPTH_TEST );

return(0); }
```

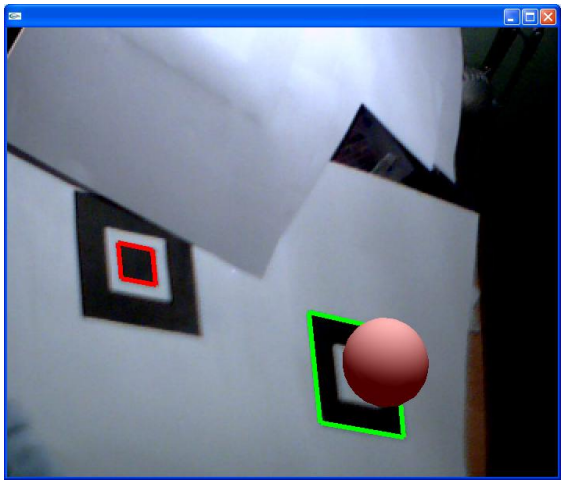
U `draw` funkciji poziva se funkcija `draw_object` kojoj se kao parametar šalje identifikacijski broj objekta i parametri transformacije. U `draw_object` funkciji definirane su postavke osvijetljena i boje za svaki objekt te se s ispitivanjem identifikacijskog broja objekta utvrđuje što će se iscrtati:

```
if(obj_id == 0){
            glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash_collide);
            glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient_collide);
            /* iscrtaj kuglu */
            glTranslatef( 0.0, 0.0, 30.0 );
            glutSolidSphere(30,20,16);

      }
      else {
            glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash);
            glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
            /* iscrtaj kocku */
            glTranslatef( 0.0, 0.0, 30.0 );
            glutSolidCube(60);

      }
```

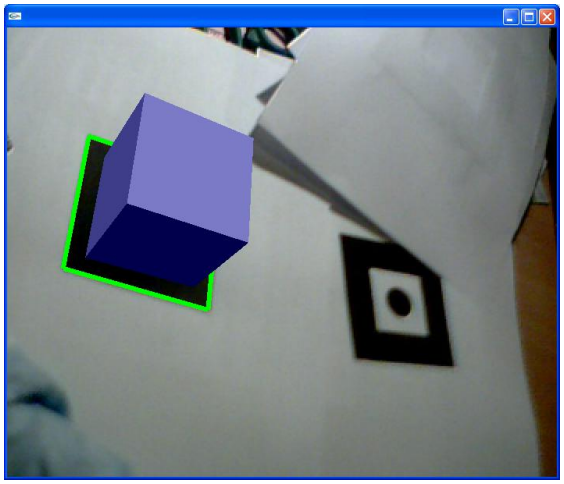
Rezultat izvođenja programa:



Slika 24. Virtualna kocka i kugla u sceni



Slika 25. Nepotpuno vidljiv marker kocke



Slika 26. Nepotpuno vidljiv marker kugle

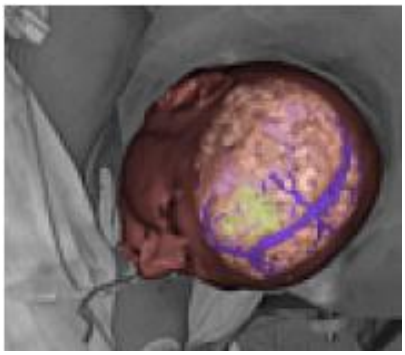
7. Primjene proširene stvarnosti

Proširena stvarnost se kao ideja pojavila još 60-tih godina dvadesetog stoljeća, a tek se 90-tih godina počelo intenzivnije raditi na izvedbi sustava proširene stvarnosti te ona danas još uvijek predstavlja relativno novo područje. Kako tehnologija još uvijek nije dovoljno razvijena za široku uporabu, većina primjena još je u eksperimentalnoj fazi.

Moguća područja primjene proširene stvarnosti su:

- Medicina

  Proširena stvarnost se ostvaruje tako da se medicinske slike preklapaju s pacijentom, čime se dobiva vrsta virtualnog rendgena u stvarnom vremenu. Dobiva se takav efekt da liječnik vidi organe pacijenta kao da je tijelo prozirno. Najčešće se primjenjuje u kirurgiji, prilikom planiranja ili izvedbe zahvata. Budući da je za ostvarenje takvih primjena proširene stvarnosti potrebna izuzetna preciznost, one za sada još nisu u širokoj uporabi.

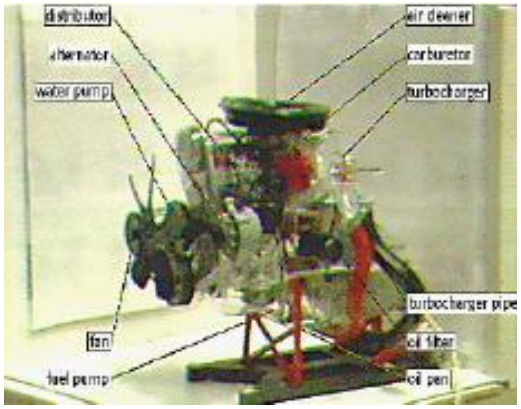


Slika 27. Slika mozga kao pomoć kirurgu

Slika 28. Slika prsnog koša kao pomoć kirurgu

- Proizvodnja i održavanje

Jedan od primjera primjene proširene stvarnosti u procesu proizvodnje jest da se vizualne instrukcije prikazuju izravno na opremi ili strojevima, te da operater ne mora proučavati upute jer ima sve potrebne informacije na pravom mjestu i pravo vrijeme. Slika 29. prikazuje motor s virtualnim oznakama dijelova. Kada korisnik obilazi motor oznake s nazivom tog dijela i osnovnom funkcijom pojavljuje na svim vidljivim dijelovima.



Slika 29. Motor s virtualnim oznakama

- Arhitektura

Proširena stvarnost se može iskoristiti u dizajnu interijera, te vizualizaciji instalacija ili struktura. Na slici 30. prikazana je prostorija u kojoj su na određenim pozicijama postavljeni markeri.

Slika 30. Prostorija sa markerima

Na slici 31. prikazani su 3D virtualni modeli zidova, vrata i namještaja.

Slika 31. Virtualni modeli namještaja

Slika 32. prikazuje kombinaciju ovih dviju scena, gdje su virtualni 3D objekti iscrtani na pozicijama markera.

Slika 32. 3D scena uklopljena u realni svijet

- Komercijalne primjene i zabava

Proširena stvarnost u uporabi je na televiziji gdje se u televizijske slike u stvarnom vremenu dodaju dodatne informacije ili reklame. Npr. u prijenos sportskih događanja mogu se ubaciti informacije o natjecateljima ili samom natjecanju (crta cilja i sl.). Također koristi se i virtualne reklame tako da se u sliku umjesto stvarnog reklamnog panoa ubaci računalno generirana reklama.

Slika 33. Proširena stvarnost u snimci plivačkog natjecanja

Sve češće se na internetu pojavljuju razne aplikacije proširene stvarnosti koje pojedine tvrtke koriste kako bi korisnici mogli isprobati njihove proizvode putem računala, npr. isprobavanje naočala ili sata putem web kamere. Na slici 34. prikazan je isječak iz snimke s aplikacije koja korisniku omogućuje isprobavanje satova tako da isprinta papirnati sat kojeg omota oko ruke i kojeg onda aplikacija prepoznaje kao marker te iscrtava na tom mjestu određeni model sata.

Slika 34. Isprobavanje sata pomoću proširene stvarnosti

Danas je sve popularnija primjena proširene stvarnosti na pokretnim uređajima. Najčešće su koristene aplikacije koje omogućuju da korisnik putem pokretnog uređaja dobiva dodatne informacije o prostoru u kojem se nalazi, npr. informacije o građevini ispred koje se nalazi, te obližnjim uslužnim objektima poput hotela, banaka, restorana i sl.

Slika 35. Proširena stvarnost na pokretnim uređajima

8. Zaključak

Kombinacija virtualnih i stvarnih scena osnovna je ideja proširene stvarnosti. Proširena stvarnost omogućuje izravan pristup informacijama tako da one budu prikazane u vidokrugu korisnika i isprepletene sa stvarnim svijetom. Proširena stvarnost je tehnologija o kojoj se u posljednje vrijeme sve više priča i čiji se pravi napredak i popularizacija tek očekuje. Razvojem ove tehnologije ostvaruju se brojne i raznovrsne mogućnosti čije se područje primjene proteže od znanstvenog do komercijalnog. Možda će upravo razvoj proširene stvarnosti u budućnosti još više pomaknuti granice dostupnosti informacija i omogućiti da proširena stvarnost postane još jedna u nizu tehnologija koje su odigrale značajnu ulogu u promijeni načina života ljudi.

9. Literatura

[1] I. S. Pandžić “Virtualna okruženja: Računalna grafika u stvarnom vremenu I njene primjene”, Element, Zagreb, 2004.

[2] I. S. Pandžić, Virtualna okruženja – laboratorijske vježbe: Proširena stvarnost, FER, http://161.53.72.23/_download/repository/VO-V4-upute%5B1%5D.pdf , 27. veljače 2010.

[3] Philip Lamb, ARToolKit Home Page, http://www.hitl.washington.edu/artoolkit/ , 27. veljače 2010.

[4] Augmented reality – Wikipedia, http://en.wikipedia.org/wiki/Augmented_reality , 27. veljače 2010.

[5] Virtual Worldlets Network – AR Development AR Development > NyARToolKit, http://www.virtualworldlets.net/Resources/Hosted/Resource.php?Name=NyARToolKit , 11. travnja 2010.

[6] Virtual Worldlets Network – AR Development AR Development > FLARToolKit, http://www.virtualworldlets.net/Resources/Hosted/Resource.php?Name=FLARToolKit , 11. travnja 2010.

[7] OpenGL Programming Guide or 'The Red Book', Addison-Wesley Publishing Company, http://fly.srk.fer.hr/~unreal/theredbook/, 28. travnja 2010.

[8] ARToolworks Support Library, http://www.artoolworks.com/support/library/Main_Page , 29. travnja 2010.

[9] Kato H., Billinghurst M., Poupyrev I., ARToolKit version 2.33, studeni 2000., http://www.tinmith.net/lca2004/ARToolkit/ARToolKit2.33doc.pdf, 22.svibnja 2010.

[10] Image Processing and the ARToolkit, http://comptuicomputervision.blogspot.com/ , 31. svibnja 2010.

[11] Augmented Reality for Interior Design, www.wimeck.com/tutorials/ar/interior/index.html, 7.lipnja 2010.

[12] Kato Hirokazo, Inside ARToolKit, http://www.hitl.washington.edu/artoolkit/Papers/ART02-Tutorial.pdf , 8. lipnja 2010.

10. Sažetak

U ovom radu su teorijski objašnjene i praktično ostvarene kombinacije virtualnih i stvarnih scena u okviru proširene stvarnosti. Definirane su i opisane osnovne karakteristike proširene stvarnosti te postupci potrebni za njenu realizaciju kao što su miješanje slike, 3D poravnavanje i metode praćenja objekata. Navedeni su najpoznatiji programski alati za izradu proširene stvarnosti i opisani principi njihovog rada. Za izradu primjera kombinacija virtualnih scena korišten je alat ARToolKit, programski jezik C++, grafičko programsko sučelje OpenGL i web kamera. Detaljnije su opisani najvažniji dijelovi programskog kôda koji odgovaraju koracima algoritma realizacije ARToolKit aplikacija. Ostvarenim primjerima naglašene su mogućnosti uključivanja raznolikih virtualnih 3D objekata te mogućnosti njihovih transformacija u stvarnoj sceni. Na kraju je dan kratak pregled osnovnih područja primjene tehnologija proširene stvarnosti.