Univerzitet u Novom Sadu

Prirodno-matematički fakultet

Departman za matematiku i informatiku

Primena Silverlight-a za prikazivanje geometrijskih tela

Master rad

Mentor:
dr Đorđe Herceg

Student:
Vesna Rvović
br. indeksa 70m/11

Novi Sad, 2014.

Sadržaj:

1. Uvod

Nastavna sredstva su didaktički oblikovana izvorna stvarnost [7] (modeli tela, udžbenici, slike...), a nastavna pomagala su oruđa za rad (pano, projektor, kalkulator...), koje nastavnik koristi prilikom podučavanja. Učenje je uspešnije ako se zasniva na više komponenti za primanje informacija. Psihološka istraživanja pokazuju da se 80% informacija prima vizualnom komponentom, pa je za učenje matematike ona najvažnija [23]. Učenje 3D geometrije, pogotovo u osnovnoj školi, za učenike je nemoguće bez vizualizacije. Koristeći računare i specijalizovane programe, olakšava se prikazivanje preciznih slika, modela, atraktivnih prezentacija...

Predavač treba da se koristi različitim nastavnim sredstvima i pomagalima kako bi nastavu održao živom, zanimljivom. Od nastavnika se očekuje da nastavu prilagodi učenicima i savremenom dobu. U današnje vreme, kada smo zahvaćeni vrtlogom tehnoloških inovacija, koje velikom brzinom postaju neizbežan deo svakodnevnice, teško je održati korak. Na neki način, ovaj rad je moj pokušaj da, kao budući nastavnik matematike, kreiram softver koji bi osavremenio nastavu, učinio je zanimljivijom, pristupačnijom i razumljivijom.

Zadatak ovog master rada je razvoj aplikacije za vizualizaciju geometrijskih tela u Silverlight tehnologiji. U radu se razmatraju matematičke osnove triangulacije poligona, predstavljanje površi u trodimenzionalnom prostoru i algoritmi za generisanje mreže tela. Na platformi Silverlight, verzija 5, implementirana je aplikacija koja omogućava kreiranje, prikaz i izmenu trodimenzionalnih geometrijskih tela. U radu je implementiran i netrivijalan algoritam za generisanje sfere.

Rad [16] koji se bavi optimalnom triangulacijom sfere, imao je najveći uticaj pri kreiranju algoritma triangulacije sfere, koji će biti prikazan u ovom radu. Knjiga [17] je neophodna za razumevanje rada [16]. U njoj su, između ostalog, definisane krivine površi i dat je dokaz o konstantnim krivinama sfere. Sa stanovišta programiranja, od najvećeg značaja za mene prilikom istraživanja za potrebe ovog rada bila je knjiga [14], jer nisam imala iskustva u programiranju i bez nje ne bih uspela da sva stečena znanja primenim, što je i bio cilj.

Implementacija softvera za generisanje i prikaz trodimenzionalnih tela, opisana u ovom radu, može da posluži kao vodič nastavnicima koji žele i sami da se oprobaju u razvoju geometrijskog softvera.

„ *Ako se već toliko zaklinjemo da nam je od svega važnije aktivno učešće dece u nastavi, ako nam je zaista iskrena ta naša želja da deca misle, da više razumevaju, a manje pamte, moramo tražiti konkretne i efikasne načine da decu pokrenemo, zainteresujemo i aktiviramo.*“

Duško Radović - „ *Na ostrvu pisaćeg stola*“.

2.Obrazovni softver i vizualizacija geometrijskih objekata

Opšte je poznata činjenica da veliki broj učenika nema interes za geometriju, a znanje ovog predmeta nalazi se na nedopustivo niskom nivou. O tome govore nastavnici, profesori fakulteta, roditelji i sami učenici. Nije tajna da se razvoj geometrijskih veština može smatrati važnim faktorom koji omogućuje spremnost čoveka za permanentno obrazovanje i samoobrazovanje u najrazličitijim oblastima ljudske delatnosti. U nastavi geometrije moramo kod učenika uporno da težimo razvoju intuicije, prostornog i logičkog mišljenja i formiranju njihovih konstruktivno - geometrijskih umeća i navika.

Nastava geometrije je značajna sa različitih gledišta [3]:

1. logičkog - izučavanje geometrije je izvor i sredstvo aktivnog intelektualnog razvoja čoveka i njegovih umnih sposobnosti;
2. saznajnog - pomoću geometrije dete spoznaje svet koji ga okružuje, njegove prostorne i količinske odnose;
3. primenjenog - trodimenzionalna euklidska geometrija je ona osnova koja obezbeđuje čovekovu spremnost za savladavanje kako bliskih oblasti, tako i mnogih profesija, čini mu dostupnim neprekidno obrazovanje i samoobrazovanje;
4. istorijskog - na primerima iz istorije razvoja geometrije prati se ne samo razvoj matematike već i ljudske kulture u celini;
5. filozofskog - geometrija pomaže da se osmisli svet u kome živimo, da se kod čoveka formiraju razvojne naučne predstave o realnom fizičkom prostoru.

Sposobnost prostorne vizualizacije nije urođena i ona zahteva učenje i vežbanje pa samim tim i podučavanje, a nepoznavanje bazičnih geometrijskih činjenica jedan je od vidova nepismenosti. Ljudskom biću nije jednostavno da prepoznaje trodimenzionalnost objekata ne samo na dvodimenzionalnim slikama nego i kad posmatra realne 3D objekte oko sebe. Vizualne informacije čovek prima putem očiju koje su u osnovi dvodimenzionalni detektori, osetljivi da detektuju dvodimenzionalnu informaciju. Stoga je uvek neophodno izabrati dobar pravac posmatranja da bi se trodimenzionalnost objekta pravilno shvatila [5].
Sledi da je geometrija u ravni jednostavnija za shvatanje od geometrije u prostoru.

Pod pojmom obrazovni računarski softver, podrazumevaju se kako gotovi računarski programi, koji se mogu koristiti u okviru sadržaja nastave, tako i programi koji pomažu i usmeravaju individualnu fazu učenja [2].

Savremena nastava matematike sve više stavlja naglasak na razumevanje matematičkih sadržaja koji su velikim delom apstraktni, što otežava njihovo razumevanje. Računar može pomoći u njihovoj vizualizaciji. **Dok objekti iz realnog života dolaskom na ekran postaju**

apstraktni, matematički objekti koji su apstraktni na ekranu postaju konkretni [6]. U tu svrhu preporučuje se korišćenje specijalizovane softverske podrške za nastavu matematike koja podržava jedan ili više matematičkih prikaza (grafičkih, simboličkih, tabelarnih). Vizualizaciju matematičkih objekata na nivou osnovnoškolske i srednjoškolske matematike jednostavno je postići programima dinamične geometrije poput GeoGebre, The Geometer's Sketchpad-a, Cabri Geometry i sl. Oni omogućavaju brzo i jednostavno predstavljanje geometrijskih sadržaja, čuvaju odnose geometrijskih objekata, omogućavaju da se matematičkim objektima lako pridruži, demonstrira ili otkrije neko svojstvo. Neki od njih poput **GeoGebre** sadrže i mogućnost da geometrijskim objektima pridružuju i algebarski prikaz i obratno. GeoGebra je besplatni multiplatformski softver za dinamičku matematiku za sve nivoe obrazovanja koji spaja geometriju, algebru, tabele, crtanje grafika, statistiku i matematičku analizu u jedan program koji se lako koristi, može se integrisati u HTML stranice. GeoGebra podržava dvodimenzionalnu geometriju, a trenutno je u razvoju verzija 5, koja će podržavati i trodimenzionalnu geometriju.
Mathematica je softver koji se koristi u raznim poljima nauke kao što su inženjerstvo i matematika. Stephen Wolfram je tvorac ovog softvera, a razvoj paketa nastavila je firma Wolfram Research. Mathematica se u matematici koristi na mnoge različite načine - kao pomoć prilikom istraživačkog rada, za različite numeričke proračune, ali i kao samostalan programski jezik koji ima mogućnost povezivanja sa drugim programskim jezicima i programskim paketima, kao što su C, .NET, JAVA, SQL, OpenOffice, Acrobat, itd. Neke od bitnih svojstava Mathematica-e su:

- Biblioteka elementarnih matematičkih funkcija;
- Biblioteka posebnih matematičkih funkcija;
- Alati za manipulaciju matricama;
- Podrška za kompleksne brojeve;
- 2D i 3D vizualizacija podataka i funkcija;
- Izračunavanje sistema jednačina, ODJ, PDJ ;
- Numerički i simbolički alati za diskretni i neprekidni račun;
- Statističke biblioteke;
- Programski jezik koji podržava proceduralne, funkcionalne i objektno-orijentisane konstrukcije;
- Alati za obradu slike;
- Alati za vizualizaciju i analizu grafova;
- Alati za kombinatoriku;

Zbog sveobuhvatnosti Mathematica nije jednostavna za korišćenje. Najveći deo korisnika ovog paketa jesu profesionalci u tehničkim naukama i matematici.

Za samu nastavu zanimljiva su posebna izdanja Mathematica Teacher's Edition i Mathematica for the Classroom. WebMathematica je poseban programski modul koji omogućava izvršavanje programa putem Interneta i rešavanje zadataka on-line. Postoje mnogi web sajtovi sa primerima u Mathematici, kao što su [21] i [22].

Poslednjih godina svedoci smo izuzetnog razvoja i popularnosti različitih vrsta mobilnih uređaja.

Osim telefoniranja, mobilni telefoni često nude i dodatne servise kao što su SMS, e-mail, pristup Internetu, pregled multimedije, itd.

Ovo opravdava razvoj geometrijskog softvera za Windows i Windows Phone, jer je očigledno da je koristan i potreban. Web pregledači, Windows Phone i Windows 8 tableti i računari prirodno rade na Silverlight-u, i iz tog razloga, za potrebe ovog rada, razvijen je program na ovoj platformi. Geometrijski programi su uglavnom samostojeći proizvodi koji se ne mogu integrisati u neki drugi obrazovni softver, za razliku od kreiranog programa. Za 3D je korišćena XNA tehnologija, koja omogućava pristup 3D hardveru i podržana je na nabrojanim platformama. Razvijene klase služe za generisanje geometrijskih objekata u zavisnosti od zadatih parametara (visina i poluprečnik osnove kod valjka i kupe, poluprečnik kod lopte). Kreirana Silverlight aplikacija predstavlja demonstraciju upotrebe tih klasa. Konačni rezultat rada jesu razvijene fleksibilne softverske komponente koje se mogu ugrađivati u druge obrazovne programe i program koji demonstrira kako se mogu koristiti.

3. Matematičke osnove

Za programiranje je neophodno poznavanje matematike, pa tako i za generisanje i prikazivanje trodimenzionalnih mreža. Potrebno je vladati i sposobnošću apstrakcije, algoritamskog mišljenja, struktuiranjem modela problema i algoritma. U tekstu koji sledi biće obrađeni matematički pojmovi koji bi trebalo da dovedu do razumevanja kreiranih algoritama.

3.1.Triangulacija poligona

Definicija 1. Triangulacija poligona je razbijanje unutrašnjosti poligona (u smislu da se on prikazuje kao unija svojih delova pri čemu svaka dva imaju disjunktne unutrašnjosti) na trouglove unutrašnjim dijagonalama koje, ako imaju zajedničkih tačaka, one mogu biti samo temena poligona.

Lema 1. Svaki poligon, sem trougla, ima unutrašnju dijagonalu.

Dokaz. Neka je dat poligon $A_1A_2A_3 \dots A_n$ i neka je A_k teme poligona sa najmanjom prvom koordinatom u zadatom desnom Dekartovom pravouglom sistemu, a ako takvih ima više, određenosti radi, uzimamo ono teme sa najmanjom drugom koordinatom. Unutrašnji ugao kod tako izabranog temena je konveksan. Njegova susedna temena poligona označimo sa A_{k-1} i A_{k+1}. Ako je dijagonala $A_{k-1}A_{k+1}$ unutrašnja, dokazali smo, a ako nije, onda postoji jedno ili više temena poligona unutar trougla $A_{k-1}A_kA_{k+1}$ ili na dijagonali $A_{k-1}A_{k+1}$. Neka je A_m takvo teme poligona koje je najdalje od dijagonale $A_{k-1}A_{k+1}$. Dijagonala A_kA_m ne seče ni jednu stranicu poligona. U suprotnom, takva stranica bi imala teme unutar trougla $A_{k-1}A_kA_{k+1}$ koje je dalje od dijagonale $A_{k-1}A_{k+1}$ nego A_m. Da je dijagonala A_kA_m unutrašnja sledi iz pretpostavke da je unutrašnji ugao poligona kod temena A_k konveksan. ■

Teorema 1. Svaki poligon ima triangulaciju. Svaka triangulacija poligona sa n temena ima tačno $n-2$ trougla.

Dokaz. Ova teorema se dokazuje potpunom matematičkom indukcijom po n- broju temena poligona $A_1A_2A_3 \dots A_n$. Ako je $n=3$, poligon je trougao i teorema važi. Neka je $n>3$ i pretpostavimo da teorema važi za svaki m-tougao gde je $m<n$.

Na osnovu Leme 1. svaki poligon, sem trougla, ima unutrašnju dijagonalu, pa unutrašnja dijagonala razbija poligon $A_1A_2A_3 \dots A_n$ na dva poligona sa brojem temena manjim od n. Po induktivnoj hipotezi ta dva poligona imaju triangulacije, što nam daje i triangulaciju polaznog poligona.

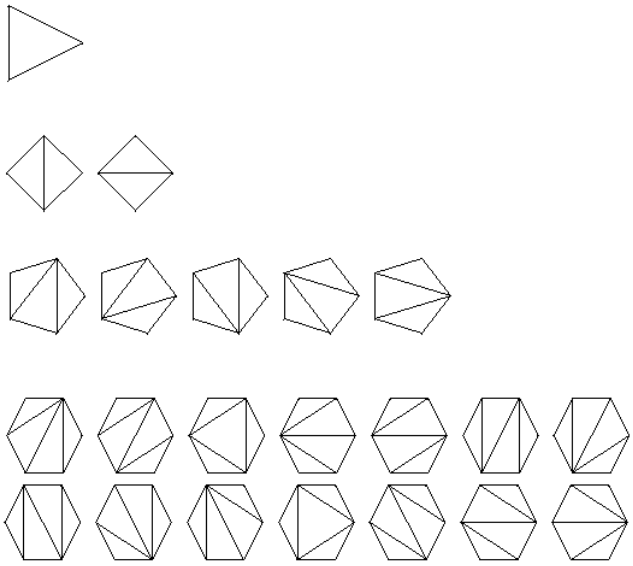
Neka je sada data proizvoljna triagulacija poligona $A_1A_2A_3 \dots A_n$ i neka je d neka unutrašnja dijagonala te triangulacije. Ona razbija poligon na dva poligona sa m_1, odnosno m_2 temena i važi $m_1+m_2 = n+2$. Po pretpostavci matematičke indukcije ta dva poligona su triangulacijom razbijena na m_1-2, odnosno m_2-2 trougla, pa je poligon razbijen na

$$(m_1-2)+(m_2-2) = m_1+m_2-4 = n+2-4 = n-2$$

trougla. ■

Teorema 2. Broj mogućih triangulacija konveksnog poligona sa n temena, u oznaci T_n $(n>2)$, zadovoljava rekurentnu formulu $T_n = \sum_{k=2}^{n-1} T_k T_{n-k+1}$, pri čemu se uzima da je $T_2 = 1$ po definiciji.

Dokaz. Za $n=3$, poligon je trougao, pa je $T_3 = 1$. Četvorougao se može triangulisati samo na dva načina, jer ima samo dve dijagonale i obe razbijaju četvorougao na dva trougla, tj. $T_4 = 2$ (Slika 1).



Slika 1. Primeri triangulacija nekih konveksnih poligona

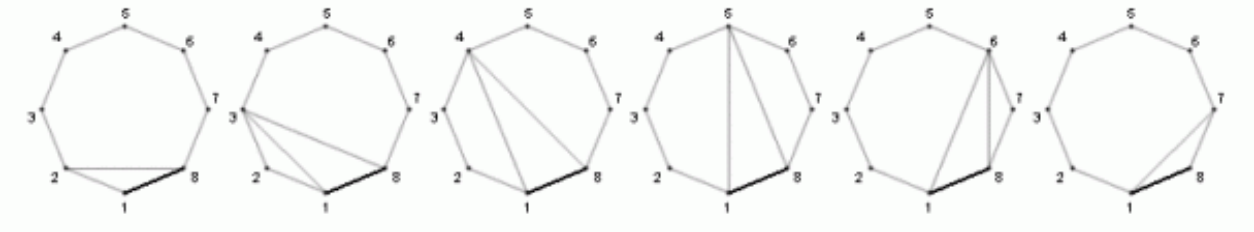
Primetimo da je svaka stranica proizvoljnog konveksnog poligona stranica tačno jednog trougla triangulacije. Označimo temena poligona brojevima 1,2,3, … , n (Slika 2) i fiksirajmo stranicu $1n$. Tada se poligon $12\dots n$ sa dijagonalama $1k$ i nk ($3 \le k \le n-2$) razbija na tri dela:

1. trougao $1nk$ čiji je broj triangulacija jedan;
2. konveksan poligon sa $n-k+1$ temena čija su temena veća ili jednaka k (koji se može triangulirati na T_{n-k+1} načina) ;
3. konveksan poligon sa k temena čija su temena manja ili jednaka k (T_k načina).

Pošto su izbori triangulacije prethodna tri poligona međusobno nezavisni, važi princip proizvoda, pa je za izabrano k-to teme broj triangulacija

$$T_k T_{n-k+1}. \qquad (1)$$

Za $k = 2$, poligon se stranicom $n2$ razbija na trougao $12n$ ($T_3 = T_2 = 1$) i konveksan poligon $234\ldots n$ sa $n-1$ temena (koji se može triangulirati na $T_{n-2+1} = T_{n-1}$ načina), a za $k = n-1$ se stranicom $1(n-1)$ razbija na trougao $1n(n-1)$ ($T_{n-(n-1)+1} = T_2 = 1$) i konveksan poligon sa $n-1$ temena (koji se može triangulirati na T_{n-1} načina). Stoga, sledi da i za ovako izabrano k važi (1).



Slika 2. Fiksiramo jednu stranicu i nad njom konstruišemo trouglove

Izbor k-tog temena možemo učiniti na više nezavisnih načina, pa preostaje sumiranje po svim mogućim vrednostima k,

$$\sum_{k=2}^{n-1} T_k T_{n-k+1}.$$

■

Za ove brojeve važi $T_{n+2} = C_n = \frac{1}{n+1}\binom{2n}{n}$, gde su C_n takozvani Katalanovi brojevi [18] .

Primer mogućeg algoritma za triangulaciju konveksnog poligona bi mogao biti sledeći:

1. Neka je $A_0, A_1, A_2, \ldots, A_{n-1}$ dati niz temena konveksnog poligona.
2. Jedna od mogućih triangulacija je data nizom trouglova $T_0, T_1, T_2, \ldots, T_{n-3}$, gde je $T_j = A_0 A_{j+1} A_{j+2}, j = 0,1, \ldots, n-3$.

Triangulacija se koristi u dokazima nekih teorema kombinatorne geometrije kao što su: razloživa jednakost poligona i Pikova teorema [15].

Pod triangulacijom poliedra podrazumevamo triagulacije svih poligona koji predstavljaju strane tog poliedra.

3.2. Generisanje triangulacija na sferi

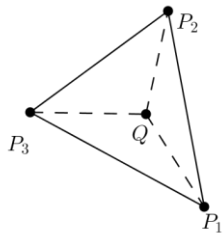
Ovo poglavlje je posvećeno obradi rada [8]. Sada se bavimo razbijanjem sfere na sferne trouglove, pri čemu je ukupan broj temena N i svaka stranica trougla (luk velike kružnice) pripada rubu tačno dva trougla, dok dva različita trougla imaju najviše jednu zajedničku stranicu. Primetimo da se svaki konveksan poliedar može projektovati na dovoljno veliku sferu sa centrom u unutrašnjosti tog poliedra. Ako su sve strane poliedra trougaone, dobijamo jednu takozvanu triangulaciju sfere. Ovo razbijanje sfere na trouglove određuje prost planaran graf sa N čvorova kod koga su sve oblasti trougaone (konture C_3) i stepen svakog čvora bar 3. Naime, prilikom projektovanja grafa sa sfere na tangentnu ravan iz tačke sfere koja je dijametralno suprotna od tačke dodira sfere i tangentne ravni, dobija se ravan graf. U nastavku se bavimo algoritmom za nalaženje svih neizomorfnih triangulacija sa N čvorova, tj. prebrojavanjem svih planarnih grafova sa N čvorova čije su sve konture C_3 i stepen svakog čvora bar 3. Prvo se predstavlja algoritam generisanja triangulacije sa N čvorova od triangulacije sa $N-1$ čvorova, a zatim algoritam za pronalaženje izomorfnih triangulacija.

Jasno je da postoji samo jedna triangulacija sfere sa 4 i 5 čvorova. Neka je T triangulacija sa $N \geq 5$ čvorova, E grana i F trouglova. Neka je X_k broj čvorova triangulacije koji su stepena k. Tada je $3F = 2E$, jer svaki trougao sadrži tri grane i svaka grana je incidentna sa tačno dva trougla. Važi i $2E = \sum kX_k$, jer je svaka grana incidentna sa dva čvora. Otuda, $6F - 6E = -2E = -\sum kX_k$. Na osnovu Ojlerove teoreme za planarne grafove sledi da je

$$12 = 6N + 6F - 6E = 6N - \sum kX_k = \sum X_k(6-k). \qquad (2)$$

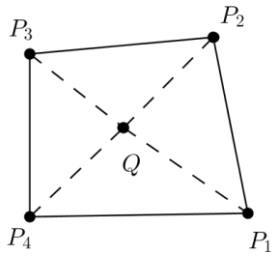
Na osnovu dobijenog izraza (2) sledi da triangulacija T mora sadržati barem jedan čvor čiji je stepen manji od 6, jer je $\sum X_k(6-k)$ pozitivan broj. Neka je Q čvor triangulacije T najmanjeg stepena. Stepen temena Q može biti 3, 4 ili 5.

Slučaj $v(Q) = 3$: Triangulacija T sadrži trouglove QP_1P_2, QP_1P_3 i QP_2P_3 prikazane na Slici 3. Otklanjanjem čvora Q i grana incidentnih sa Q dobija se triangulacija sa $N-1$ čvorova. Obrnutim procesom se dobija triangulacija sa N čvorova od triangulacije sa $N-1$, dodavanjem čvora Q stepena 3 i odgovarajućih grana $QP_k, k = 1,2,3$.



Slika 3. Slučaj $v(Q) = 3$

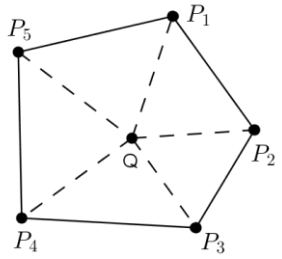
Slučaj $v(Q) = 4$: U ovom slučaju triangulacija T ima formu prikazanu na Slici 4. Na osnovu Žordanove teoreme (Q se nalazi u jednoj od dve moguce oblasti) P_1 i P_3 nisu susedni ili P_2 i P_4 nisu susedni čvorovi. Bez umanjenja opštosti, neka P_1 i P_3 nisu susedni. Tada se odstranjivanjem čvora Q i grana QP_k i dodavanjem grane P_1P_3 dobija triangulacija T' sa $N-1$ čvorova. Obrnutim procesom od T' se dobija T.



Slika 4. Slučaj $v(Q) = 4$

Slučaj $v(Q) = 5$: Tvrdimo da postoji neko P_k sa Slike 5. koje nije susedno ni sa jednim drugim P_i osim sa ona dva prikazana na toj slici .
U suprotnom, to bi važilo i za P_1, pa bi P_1 bio susedan sa P_3 ili sa P_4. Recimo da je susedan sa P_3 (grana bi pripadala oblasti koja je odredjena sa krivom $P_1P_2P_3P_4P_5$ u kojoj nije Q). Ali tada ne bi isto moglo da vazi i za P_2, tj. P_2 ne bi moglo biti susedno ni sa P_5 ni sa P_4. Kontradikcija.

Stoga, možemo pretpostaviti da P_1 nije povezano ni sa P_3 ni sa P_4. Uklanjanjem temena Q i grana QP_k i dodavanjem P_1P_3 i P_1P_4 dobija se triangulacija T' sa $N-1$ čvorovima, a obrnutim procesom od T' se dobija T.



Slika 5. Slučaj $v(Q) = 5$

Na taj način dobijamo tri operacije koje, kada se primene na sve moguće načine na sve triangulacije od $N-1$ čvorova, će dati sve moguće triangulacije od N čvorova. Postoji mogućnost da su neke od dobijenih triangulacija izomorfne, te se u nastavku rada objašnjava kako se prepoznaju izomorfne triangulacije da bi se došlo do traženog broja triangulacija.

Sada pretpostavimo da su T_1 i T_2 dve triangulacije sa po N čvorova. Trouglove triangulacije T_k poređamo redom po najvećem stepenu čvora trougla, zatim po stepenu drugog najvećeg i na kraju po stepenu trećeg čvora. Sa M_k obeležimo skup maksimalnih trouglova triangulacije T_k. Ako je

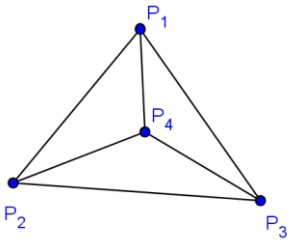
$P_1P_2P_3 \in M_2$ i $f: T_1 \longrightarrow T_2$ izomorfizam, tada izomorfizam mora sadržati $f(Q_i) = P_i$ za neko $Q_1Q_2Q_3 \in M_1$.
Pretpostavimo da f^* preslikava Q_i u P_i, $i = 1,2,3$ i želimo da proširimo f^* na izomorfizam $T_1 \longrightarrow T_2$. Krećući se u pravcu kazaljke na satu od Q_3, neka je Q_4 prvi sledeći čvor susedan sa Q_1. Pošto se radi o triangulaciji, mora postojati grana koja spaja Q_3 sa Q_4. Neka je P_4 čvor različit od P_2 tako da je $P_1P_3P_4 \in T_2$. Ako je f izomorfizam dobijen proširenjem funkcije f^*, onda je $f(Q_4) \neq f(Q_2) = P_2$ i $f(Q_4)$ je susedno sa $f(Q_1) = P_1$ i $f(Q_3) = P_3$, pa mora biti $f(Q_4) = P_4$. Nastavljajući postupak, na ovaj način uočavamo da je f definisana na svim čvorovima koji su susedni sa Q_1. Ponavljajući ovaj postupak za čvorove susedne sa Q_1 definiše se f za sva temena na udaljenosti 2 od Q_1. Indukcijom ovaj metod određuje izomorfizam f proširujući f^*, ako uopšte postoji. Primenjujući ovaj algoritam na sve moguće f^* i proveravajući svaku funkciju f da li je izomorfizam, dobijamo odgovor na pitanje da li su triangulacije T_1 i T_2 izomorfne.

Ne primenjujemo treću operaciju algoritma generisanja ako dovodi do triangulacije sa čvorom stepena 3 ili 4, a drugu operaciju ne koristimo ako dovodi do triangulacije sa čvorom stepena 3. Takve triangulacije će se dobiti primenom prethodnih operacija na drugim triangulacijama.

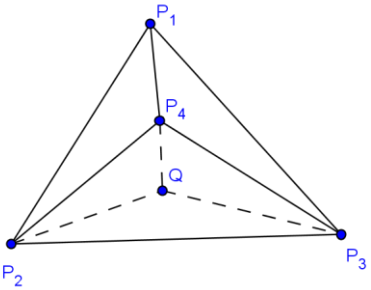
U Tabeli 1. predstavljen je broj neizomorfnih triangulacija $L(N)$ i broj triangulacija bez čvora stepena 3 $M(N)$, gde je broj čvorova triangulacije N.

Tabela 1.

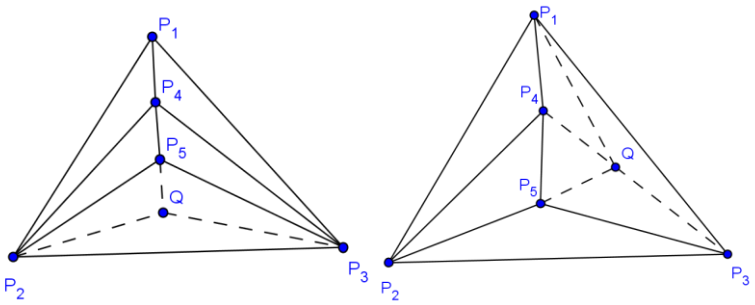
N	6	7	8	9	10	11	12
$L(N)$	2	5	14	50	233	1249	7595
$M(N)$	1	1	2	5	12	34	130



Slika 6. Triangulacija sa 4 čvora

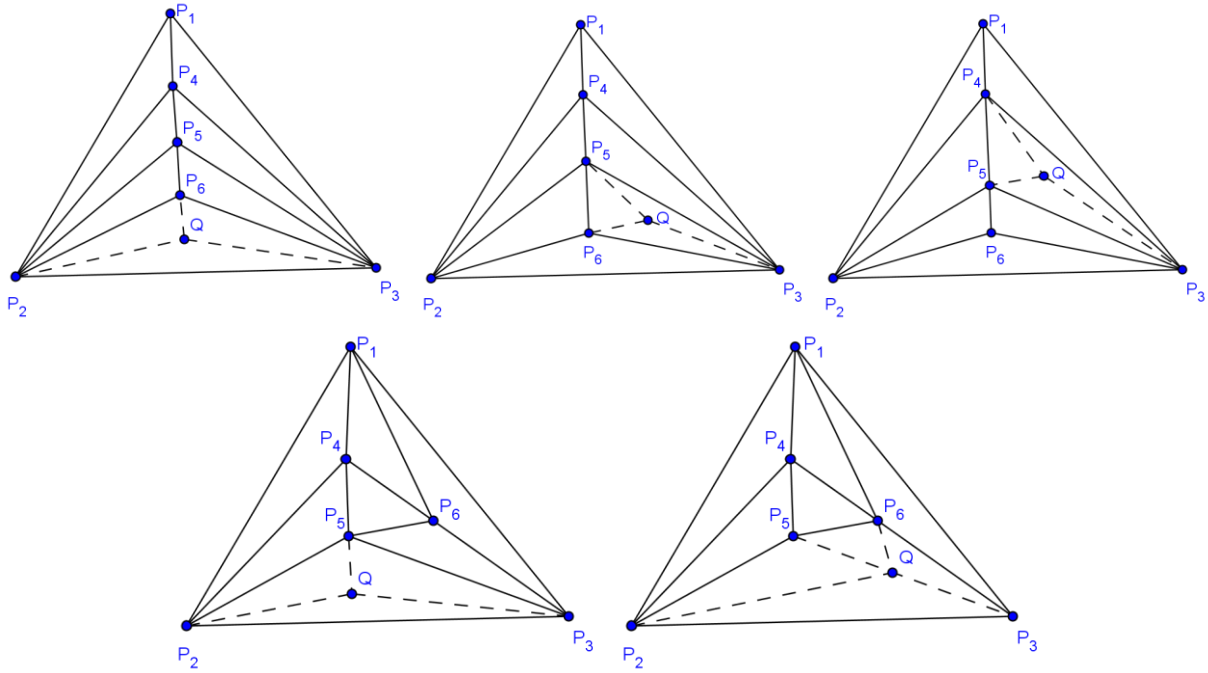


Slika 7. Triangulacije sa 5 čvorova, dobijena dodavanjem tačke Q i odgovarajućih grana na prethodnu triangulaciju



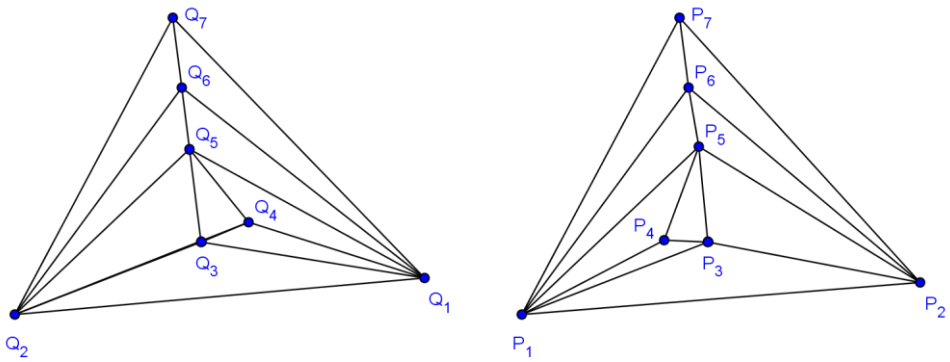
Slika 8. Kreiranje neizomorfnih triangulacija sa 6 čvorova pomoću opisanih metoda

Slikom 8. predstavljene su neizomorfne triangulacije sa 6 čvorova. Maksimalan trougao prve triangulacije je P_2P_3Q, gde su stepeni čvorova redom 5, 5 i 3, a druge su svi trouglovi maksimalni i svaki čvor je stepena 4, pa samim tim ne postoji izomorfizam između ovih triangulacija.



Slika 9. Kreiranje neizomorfnih triangulacija sa sedam čvorova

Prethodnom slikom su predstavljene sve moguće neizomorfne triangulacije sa sedam čvorova. U prvom redu su triangulacije nastale dodavanjem čvora i odgovarajućih grana na triangulaciju sa 6 čvorova koja je prikazana levo na Slici 8, a u drugom redu su triangulacije dobijene korišćenjem opisanih metoda na triangilaciju sa 6 čvorova, gde su svi čvorovi stepena 4 (Slika 8. desno)



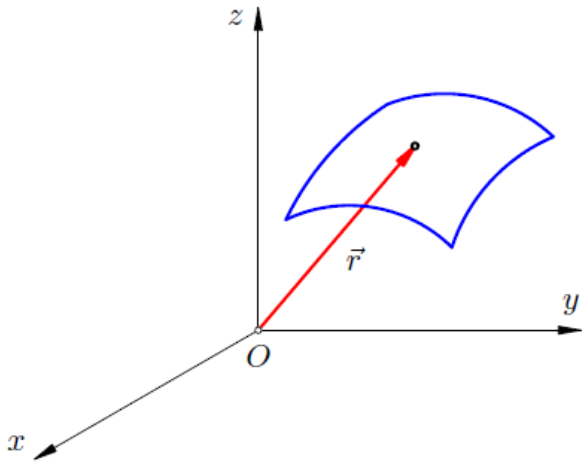
Slika 10. Primer izomorfnih triangulacija sa sedam čvorova

Slika 10. prikazuje dve izomorfne triangulacije sa 7 čvorova. $Q_1Q_2Q_3$ je maksimalan trougao triangulacije T_1, a $P_1P_2P_3$ je maksimalan trougao triangulacije T_2. Neka je $f^*(Q_i) = P_i, i = 1,2,3$. Proveravamo da li možemo proširiti funkciju f^* na izomorfizam $f: T_1 \longrightarrow T_2$ opisanom metodom. Funkcija $f(Q_i) = P_i, i = 1,2, \dots ,7$ je izomorfizam.

Inače, ravanska i sferna triangulacija se koristi u geodeziji za premer površina koristeći ravansku, odnosno sfernu trigonometriju [19].

3.3. Parametrizacija

Površi u prostoru se parametrizuju korišćenjem dva parametra, koji se najčešće označavaju sa u i v,



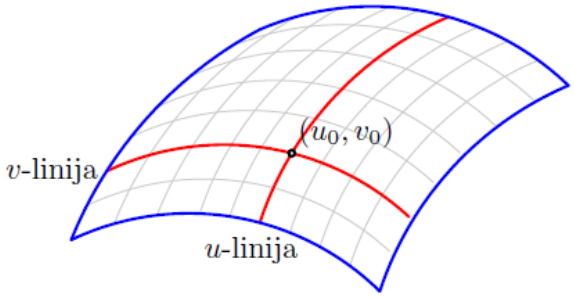
Slika 11: Parametrizacija površi

$$\vec{r} = \vec{r}(u, v).$$

To znači da se svaka od promenljivih može prikazati kao funkciju dva parametra

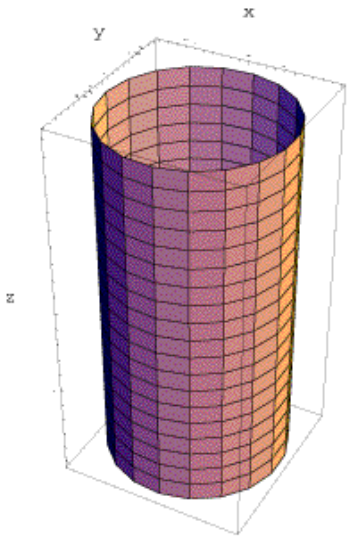
$$(x, y, z) = (x(u, v), y(u, v), z(u, v)).$$

Ako se fiksira jedan od parametara u, odnosno v, dobija se odgovarajuća kriva. Ako se postavi da je v konstantan i jednak v_0, a u se menja, dobija se u-linija, a ako se stavi da je u konstantan i jednak u_0, dobija se v-linija.



Slika 12: u i v linije površi

3.3.1. Cilindar



Slika 13. u i v linije cilindra

Parametrizacija cilindra data je sa

$$f(u, v) = (a\cos(u), a\sin(u), v),$$

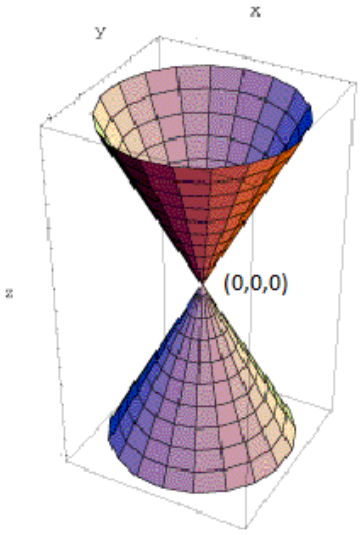
gde je $u \in [0,2\pi), v \in (-\infty, \infty)$ i $a > 0$.
v-linije su prave, a u-linije su podudarne kružnice poluprečnika a.

Valjak se može aproksimirati na jednostavan način pomoću n podudarnih pravougaonika (omotač) i dva podudarna pravilna n-tougla (baze), što znači da je ta aproksimacija valjka pravilna prizma čija je osnova pravilan n-tougao.

Algoritam:

1. Kreira se niz tačaka donje baze $A_i = \left(a\cos(i\frac{2\pi}{n}), a\sin(i\frac{2\pi}{n}), h_0\right)$, gde je $i = 0,1,\dots,n-1$.
2. Kreira se niz tačaka gornje baze $B_i = \left(a\cos(i\frac{2\pi}{n}), a\sin(i\frac{2\pi}{n}), h_1\right)$, gde je $i = 0,1,\dots,n-1$ i a poluprečnik baze i $h_0 - h_1$ visina valjka.
3. Konstruišu se pravougaonici koji aproksimiraju omotač valjka povezivanjem odgovarajućih tačaka gornje i donje baze na sledeći način: $P_0 = A_0A_1B_1B_0$, $P_1 = A_1A_2B_2B_1$,..., $P_{n-1} = A_{n-1}A_0B_0B_{n-1}$.
4. Za zadate tačke baza konstruišu se n-touglovi $P_n = A_0A_1 \dots A_{n-1}$ i $P_{n+1} = B_0B_1 \dots B_{n-1}$.

3.3.2. Konus



Slika 14. u i v linije konusa

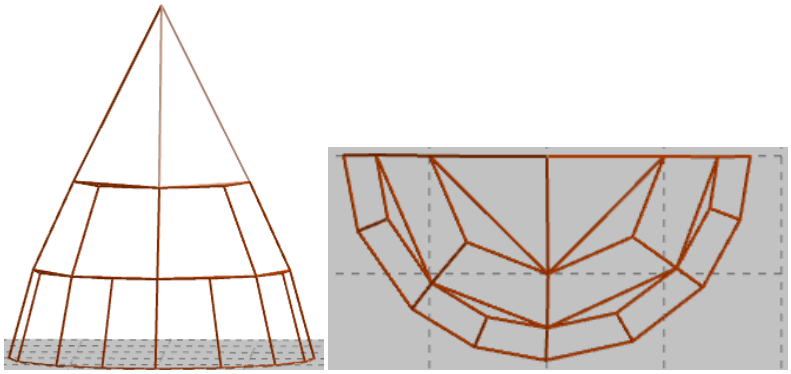
Parametrizacija konusa data je sa

$$f(u,v) = (av\cos(u), av\sin(u), bv),$$

gde je $u \in [0,2\pi), v \in (-\infty,\infty)$, i $a,b > 0$.

Kod konusa, kao i kod cilindra, v-linije su prave, a u-linije su kružnice. Razlika je u tome što su kružnice u-linije sve većeg obima kako se udaljavaju od tačke $(0,0,0)$.

Ako je potreba za aproksimacijom kupe takva da greške krivina (razlika između krivine i aproksimacije krivine) što manje zavise od udaljenosti u-linije od vrha kupe, onda se to može postići na sledeći način. Podeli se omotač kupe u-linijama, tako što se podeli visina na jednake delove. „Trake“ omeđene u -linijama se aproksimiraju četvorouglovima, tako da broj ćetvorouglova bude srazmeran udaljenosti „trake“ od vrha kupe, a sam vrh kupe (od vrha kupe do najbliže u-linije) se aproksimira podudarnim jednakokrakim trouglovima. Što se više „traka“ napravi, to će greške krivina manje zavisiti od udaljenosti od vrha kupe. Međutim, problem kod ovog načina predstavljanja kupe je što se mreža pravi iz delova koji nisu spojeni, pa se javljaju „rupe“ koje dovode do lošeg izgleda kupe (videti Sliku 15).



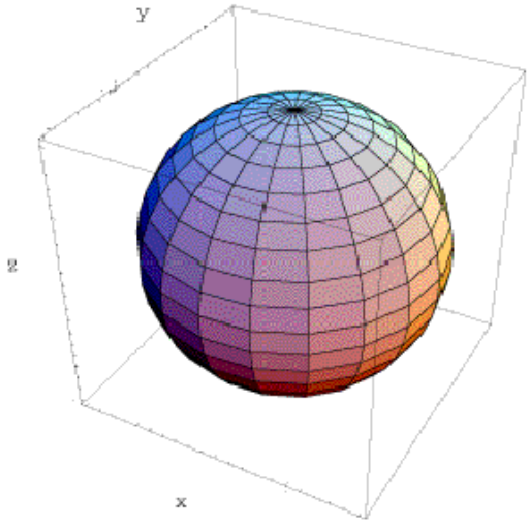
Slika 15.

Ako je jedini uslov aproksimacije izgled kupe, onda je dovoljno da se omotač kupe predstavi preko podudarnih jednakokrakih trouglova sa temenom u vrhu kupe, odnosno, da se aproksimira n-tostranom piramidom. Baza kupe se aproksimira kao i baza valjka.

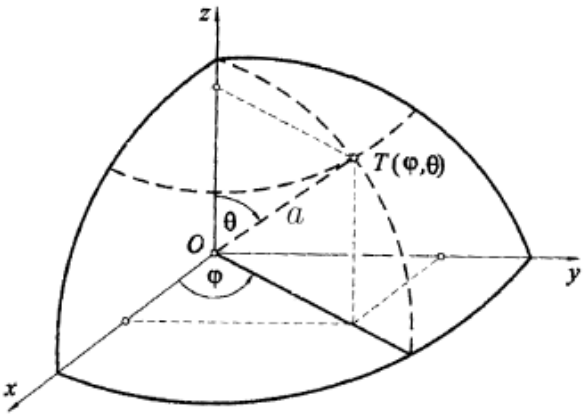
Algoritam:

1. Neka je $A_n = (0,0,h)$ tačka vrha kupe, gde je treća koordinata visina kupe.
2. Kreiraju se tačke baze na sledeći način: $A_i = \left(a\cos\left(i\frac{2\pi}{n}\right), a\sin\left(i\frac{2\pi}{n}\right), 0\right)$, $i = 0,1,\ldots,n-1$, gde je a poluprečnik osnove.
3. Prave se trouglovi koji aproksimiraju omotač $T_0 = A_0A_1A_n, T_1 = A_1A_2A_n, \ldots, T_{n-1} = A_{n-1}A_0A_n$.
4. Baza se aproksimira n-touglom $B = A_0A_1 \ldots A_{n-1}$.

3.3.4. Sfera



Slika 16. u i v linije sfere



Slika 17. Sferne koordinate

Parametrizacija sfere data je sa

$$f(u,v) = (a\cos(u)\cos(v), a\sin(u)\cos(v), a\sin(v)),$$

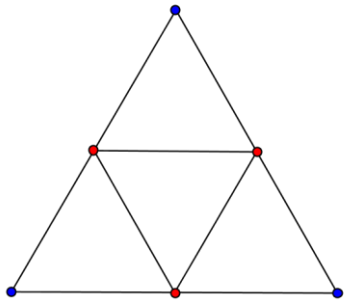
gde je $u \in [0,2\pi]$, $v \in [-\frac{\pi}{2},\frac{\pi}{2}]$ i $a > 0$ je poluprečnik.

Ukoliko se triangulacija sfere vrši preko njenih u i v linija (tradicionalna mreža preko geografske širine i dužine), u i v linije neće odstupati od mreže jednako na svim njenim delovima. Ova vrsta nepravilnosti se može umanjiti povećavanjem broja u i v linija kako se približavaju ekvatoru, ali kao i kod kupe, triangulacija će biti iz delova koji nisu spojeni, pa izgled sfere neće biti zadovoljavajući.

Krivine sfere su konstantne [17], pa je mišljenje da bi aproksimacija sfere mrežom trouglova trebalo da bude regularna (u smislu da su sva temena istog stepena i jednako udaljena od svih temena sa kojima su incidentna) opravdavajuće.

Zanimljivo je da se vremenska prognoza računa uz pomoć parcijalnih diferencijalnih jednačina (PDJ) na sferi. Naravno, zbog kompleksnosti računa, prave se programi pomoću kojih se rešavaju ove jednačine na mreži koja aproksimira sferu. Tradicionalna mreža ne daje dobre rezultate, jer nije ujednačena, pogotovo na polovima. U literaturi [16] opisano je istraživanje optimalne triangulacije sfere i imalo je najveći značaj pri kreiranju algoritma triangulacije sfere koji je ovde predstavljen.

Počinje se sa ikosaedrom upisanim u jediničnoj sferi. Od svakog trougla prave se četiri tako što se svaka središnja tačka svake ivice projektuje na sferu u pravcu normale. Ovaj proces generiše niz triangulacija koje imaju 20, 80, 320,1280, 5120, 20480, 81820,... trouglova redom.



Slika 18. Način kreiranja četiri od jednog trougla
(plave tačke označavaju tačke prethodne iteracije, a crvene novodobijene)

Algoritam:

1. Kreira se niz tačaka ikosaedra upisanog u jediničnu kružnicu, koordinate preuzete iz izvora [27]:

$T_0 = (0, 0, 1)$; T_1= $(0.894, 0, 0.447)$; $T_2 = (0.276, 0.851, 0.447)$;
$T_3 = (-0.724, 0.526, 0.447)$; T_4=$(-0.724, -0.526, 0.447)$; $T_5 = (0.276, -0.851, 0.447)$;
T_6=(0.724, 0.526, -0.447); $T_7 = (-0.276, 0.851, -0.447)$; $T_8 = (-0.894, 0.000, -0.447)$;
$T_9 = (-0.276, -0.851, -0.447)$; $T_{10} = (0.724, -0.526, -0.447)$; $T_{11} = (0, 0, -1)$

2.Zatim se kreiraju strane ikosaedra, to je niz uređenih trojki:

$F_0 = (T_0, T_2, T_1)$; $F_1 = (T_0, T_3, T_2)$; $F_2 = (T_0, T_4, T_3)$; $F_3 = (T_0, T_5, T_4)$;
$F_4 = (T_0, T_1, T_5)$; $F_5 = (T_{11}, T_6, T_7)$; $F_6 = (T_{11}, T_7, T_8)$; $F_7 = (T_{11}, T_8, T_9)$;
$F_8 = (T_{11}, T_9, T_{10})$; $F_9 = (T_{11}, T_{10}, T_6)$; $F_{10} = (T_1, T_2, T_6)$; $F_{11} = (T_2, T_3, T_7)$;
$F_{12} = (T_3, T_4, T_8)$; $F_{13} = (T_4, T_5, T_9)$; $F_{14} = (T_5, T_1, T_{10})$; $F_{15} = (T_6, T_2, T_7)$;
$F_{16} = (T_7, T_3, T_8)$; $F_{17} = (T_8, T_4, T_9)$; $F_{18} = (T_9, T_5, T_{10})$; $F_{19} = (T_{10}, T_1, T_6)$;

3. Za zadati niz trougolova $F_0, F_1, \ldots, F_{n-1}$ prave se nove tačke kao na Slici 18.

$A_i = \frac{F_{i,0}+F_{i,1}}{2}, B_i = \frac{F_{i,1}+F_{i,2}}{2}, C_i = \frac{F_{i,2}+F_{i,0}}{2}$ za svako $i \leq$ broj trouglova. $F_{i,j}$ je oznaka za j −to teme i −tog trougla , a oznaka $A = (B + C)/2$ podrazumeva da je svaka koordinata tacke A jednaka aritmetičkoj sredini odgovarajućih koordinata tačaka B i C.

4. Tačke A_i, B_i, C_i, $i \leq$ broj trouglova, projektuju se na sferu množeći koordinate recipročnom vrednošću kvadratnog korena zbira kvadrata koordinata. Neka su te nove tačke A_{is}, B_{is} i C_{is}.

5. Kreiraju se trouglovi $F_{i,0}A_{is}C_{is}$, $F_{i,1}A_{is}B_{is}$, $F_{i,2}B_{is}C_{is}$ i $A_{is}B_{is}C_i$ koji su članovi novog niza trouglova, odnosno čine telo koje bolje aproksimira sferu od prethodnog.

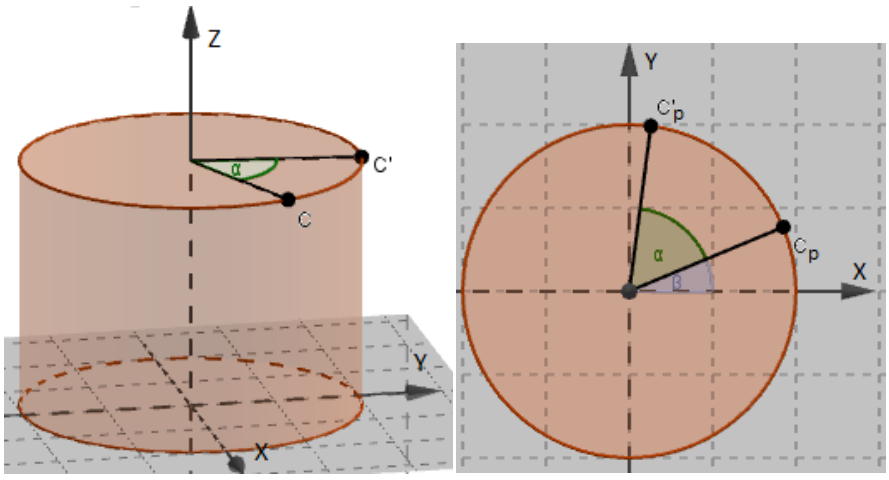
6. Nakon napravljenog novog tela, vraća se na 3. korak i tako dok se ne dobije željeni izgled sfere.

3.4. Matrice rotacije

Rotirajući objekti i virtuelne kamere su značajni u kompjuterskoj grafici. U 3D osa rotacije može biti bilo koja proizvoljna osa. Prvo će biti predstavljene matrice rotacije oko osnovnih osa, a zatim oko proizvoljne ose koja sadrži koordinatni početak.

3.4.1. Rotacija tačke oko x, y i z ose

Rotacija oko z ose



Slika 19. Levo tačke C i C', a desno njihove projekcije na xy ravan

Neka je $C = (x, y, z)$, $C' = (x', y', z')$, $\alpha = \sphericalangle C_p O C'_p$, $\beta = \sphericalangle x O C_p$, gde su C_p i C'_p projekcije tačaka redom C i C' na xy ravan. Tada je

$$x = r\cos\beta \text{ i } y = r\sin\beta,$$
$$x' = r\cos(\beta + \alpha) = r\cos\beta\cos\alpha - r\sin\beta\sin\alpha,$$
$$y' = r\sin(\beta + \alpha) = r\sin\beta\cos\alpha + r\cos\beta\sin\alpha,$$

gde je r rastojanje tačke C od z ose. Sledi

$$x' = x\cos\alpha - y\sin\alpha,$$
$$y' = y\cos\alpha + x\sin\alpha,$$
$$z' = z,$$

pa je matrica rotacije oko z ose za ugao α

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Analogno se dobijaju matrice rotacije za ugao α oko x i y ose.
Matrica rotacije oko x -ose za ugao α je

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}.$$

Matrica rotacije oko y -ose za ugao α je

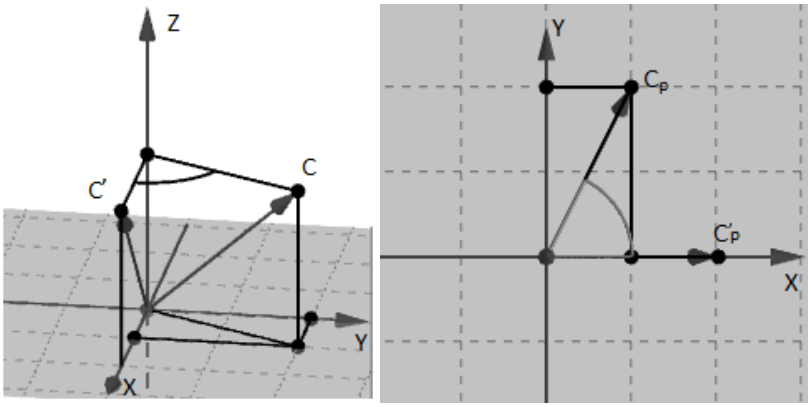
$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}.$$

3.4.2. Rotacija oko proizvoljne ose koja sadrži koordinatni početak

Problem rotacije oko proizvoljne ose u tri dimenzije javlja se u mnogim oblastima, uključujući i kompjutersku grafiku. Ovde će biti predstavljen algoritam kojim se vrši rotacija tačke oko proizvoljne ose koja sadrži koordinatni početak, jer se svaka osa može translacijom dovesti u taj položaj.

1.Osu rotacije definišemo tačkom kroz koju prolazi (koordinatni početak) i vektorom pravca $\overrightarrow{OC} = (a, b, c)$.

2. Sledeći korak je rotacija vektora pravca oko z ose za ugao α kojom se dobija vektor u xz ravni.



Slika 20.

Rotacijom oko $z-$ose vektora pravca $\overrightarrow{OC} = (a, b, c)$ tako da dobijeni vektor pripada xz ravni dobija se vektor $\overrightarrow{OC'} = (a', b', c')$, gde je $a' = \sqrt{a^2 + b^2}$, $b' = 0$ i $c' = c$.

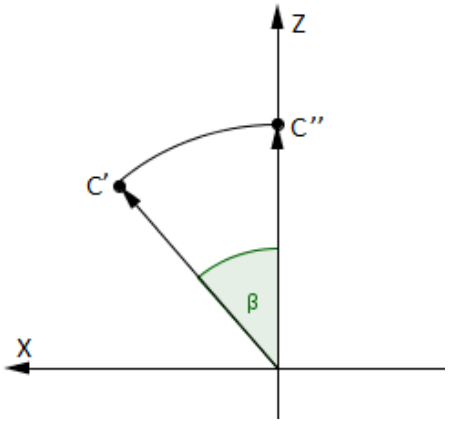
Neka je $\alpha = \sphericalangle xOC_p$, tada važi:

$$\cos\alpha = \frac{a}{\sqrt{a^2+b^2}} \text{ i } \sin\alpha = \frac{b}{\sqrt{a^2+b^2}}.$$

Na osnovu osobine sinusne i kosinusne funkcije $\cos(-\alpha) = \cos\alpha$ i $\sin(-\alpha) = -\sin\alpha$, kao i već navedene matrice rotacije oko z ose za zadati ugao, dobija se matrica transformacije vektora.

$$T_{xz} = R_z(-\alpha) = \begin{bmatrix} \frac{a}{\sqrt{a^2+b^2}} & \frac{b}{\sqrt{a^2+b^2}} & 0 \\ -\frac{b}{\sqrt{a^2+b^2}} & \frac{a}{\sqrt{a^2+b^2}} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

3. Vektor iz xz ravni se rotira oko y ose za ugao β tako da dobijeni vektor ima isti pravac i smer kao z osa.



Slika 21.

Rotacijom vektora $\overrightarrow{OC'}$ na ovaj način dobija se $\overrightarrow{OC''} = (a'', b'', c'')$, gde je $a'' = 0$, $b'' = 0$ i $c'' = \sqrt{a^2 + b^2 + c^2}$. Tada važi:

$$\cos\beta = \frac{c'}{\sqrt{a^2+b^2+c^2}} = \frac{c}{\sqrt{a^2+b^2+c^2}} \text{ i } \sin\beta = \frac{a'}{\sqrt{a^2+b^2+c^2}} = \frac{\sqrt{a^2+b^2}}{\sqrt{a^2+b^2+c^2}}.$$

Analogno kao pod 2. dobija se matrica transformacije

$$T_z = R_y(-\beta) = \begin{bmatrix} \frac{c}{\sqrt{a^2+b^2+c^2}} & 0 & -\frac{\sqrt{a^2+b^2}}{\sqrt{a^2+b^2+c^2}} \\ 0 & 1 & 0 \\ \frac{\sqrt{a^2+b^2}}{\sqrt{a^2+b^2+c^2}} & 0 & \frac{c}{\sqrt{a^2+b^2+c^2}} \end{bmatrix}.$$

4. Vrši se rotacija oko z ose za zadati ugao $R_z(\theta)$.
5. Primenjuju se inverzne transformacije redom T_z^{-1} i T_{xz}^{-1}.

Rezultat je matrica

$$T_{xz}^{-1}T_z^{-1}R_z(\theta)T_zT_{xz} =$$

$$= \begin{bmatrix} \frac{a^2 + (b^2+c^2)\cos\theta}{a^2+b^2+c^2} & \frac{ab(1-\cos\theta) - c\sqrt{a^2+b^2+c^2}\sin\theta}{a^2+b^2+c^2} & \frac{ac(1-\cos\theta) + b\sqrt{a^2+b^2+c^2}\sin\theta}{a^2+b^2+c^2} \\ \frac{ab(1-\cos\theta) + c\sqrt{a^2+b^2+c^2}\sin\theta}{a^2+b^2+c^2} & \frac{b^2 + (a^2+c^2)\cos\theta}{a^2+b^2+c^2} & \frac{bc(1-\cos\theta) - a\sqrt{a^2+b^2+c^2}\sin\theta}{a^2+b^2+c^2} \\ \frac{ac(1-\cos\theta) - b\sqrt{a^2+b^2+c^2}\sin\theta}{a^2+b^2+c^2} & \frac{bc(1-\cos\theta) + a\sqrt{a^2+b^2+c^2}\sin\theta}{a^2+b^2+c^2} & \frac{c^2 + (a^2+b^2)\cos\theta}{a^2+b^2+c^2} \end{bmatrix}$$

3.4.3. Ojlerovi uglovi

Ako su $Oxyz$ i $Ox'y'z'$ dva pravougla Dekartova sistema iste (desne) orjentacije sa istim koordinatnim početkom O, onda se iz jednog sistema transformacijama može preći u drugi sistem [20].

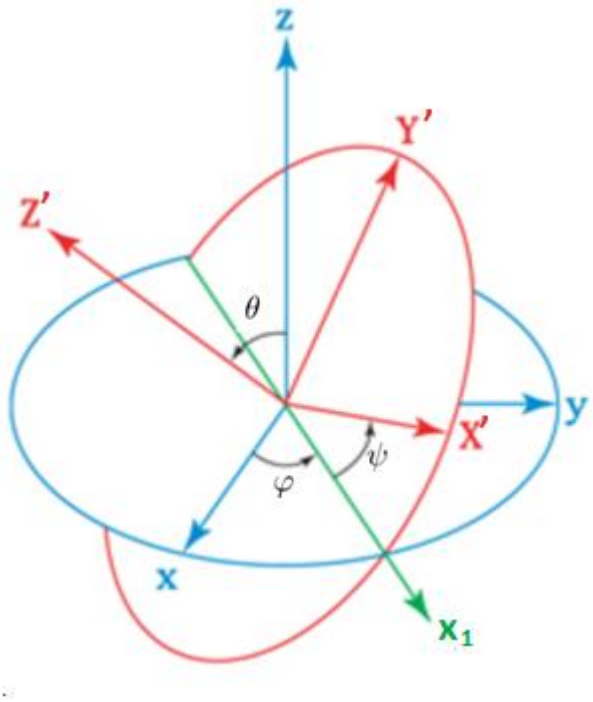
$$Oxyz \xrightarrow{\rho_\varphi^z} Ox_1y_1z \xrightarrow{\rho_\theta^{x_1}} Ox_1y_2z' \xrightarrow{\rho_\psi^{z'}} Ox'y'z',$$

gde je x_1 prava preseka ravni xy i $x'y'$, $\varphi = \sphericalangle xOx_1, \theta = \sphericalangle zOz', \psi = \sphericalangle x_1Ox'$, a transformacija predstavlja proizvod tri matrice rotacije $T = R_z(\varphi)R_x(\theta)R_z(\psi)$, pa je

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = T^{-1}\begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

$$T^{-1} = (R_z(\varphi)R_x(\theta)R_z(\psi))^{-1} = R_z(\psi)^{-1}R_x(\theta)^{-1}R_z(\varphi)^{-1},$$

$$T^{-1} = \begin{bmatrix} \cos\psi\cos\varphi - \cos\theta\sin\varphi\sin\psi & \cos\psi\sin\varphi + \cos\theta\cos\varphi\sin\psi & \sin\psi\sin\theta \\ -\sin\psi\cos\varphi - \cos\theta\sin\varphi\cos\psi & -\sin\psi\sin\varphi + \cos\theta\cos\psi\cos\varphi & \cos\psi\sin\theta \\ \sin\theta\sin\varphi & -\sin\theta\cos\varphi & \cos\theta \end{bmatrix}.$$



Slika 22. Ojlerovi uglova

Uglovi φ, θ i ψ se nazivaju Ojlerovi uglovi. Njihove vrednosti ograničene su na intervale

$$0 \leq \varphi \leq 2\pi, 0 \leq \theta \leq \pi, 0 \leq \psi \leq 2\pi.$$

4. Korišćena tehnologija

Microsoft .NET Framework je softverska platforma koja može biti instalirana na računarima koje pokreće Microsoft Windows operativni sistem. On uključuje veliki broj gotovih biblioteka za uobičajene primene u programiranju i virtuelnu mašinu koja upravlja izvršavanjem programa pisanih specijalno za .NET Framework. .NET podržava više programskih jezika (VB.NET, C#...). Za razvoj aplikacija koje rade na .NET Frameworku, potreban je i Microsoft SDK (engl. Microsoft Software Development Kit) i Visual Studio.

Bazne klase pružaju širok spektar mogućnosti, uključujući korisnički interfejs, pristup podacima, bazama, razvoj web aplikacija,... Biblioteke klasa se koriste od strane programera, koji ih kombinuju sa svojim kodom u toku izrade aplikacija.

Programi pisani za .NET Framework izvršava se u specifičnom softverskom okruženju. Naime, ovo okruženje je poznato kao Common Language Runtime (CLR). CLR obezbeđuje da programeri ne treba da razmatraju mogućnosti specifičnih procesora koji će izvršiti program. On takođe pruža druge važne usluge kao što su bezbednost, upravljanje memorijom, a i rukovanje izuzecima. Biblioteke klasa (Framework Class Library) i CLR zajedno čine .NET Framework. Za potrebe ovog rada korišćen je Microsoft .NET Framework 4.5.

XAML („*Extensible Application Markup Language*“) predstavlja deklarativni jezik koji se koristi za inicijalizaciju .NET objekata. Drugim rečima, XAML dokumenti definišu raspored panela, dugmadi i drugih kontrola koje čine UI (user interface) aplikacije.

Iako je takav slučaj moguć, XAML se ne piše „ručno“. Postoji niz alata čija uloga je generisanje XAML koda. Koji alat će biti korišćen, zavisi od uloge korisnika u razvojnom procesu aplikacije. Programeri će, naravno, koristiti Visual Studio, dok će se grafički dizajneri puno bolje snaći u Microsoft Expression Blend-u. Upravo u toj činjenici leži i jedna od najvećih prednosti XAML-a – on predstavlja tehnologiju koju će deliti programeri i dizajneri korisničkog interfejsa. Pre pojave XAML-a proces kreiranja korisničkog interfejsa, koji bi uključivao dizajnere i programere, bio je frustrirajući za obe strane i na kraju se obično svodio na sledeće: grafički dizajner bi pripremio model korisničkog interfejsa, koji bi programeri kasnije morali da prevode u kod. Upotrebom XAML-a proces može izgledati ovako: programer kreira osnovni korisnički interfejs i potom ga predaje dizajn timu koji dalje radi na njemu.

I pored saradnje sa grafičkim dizajnerima postoji mnogo drugih razloga za korišcenje XAML-a:

- XAML je koncizan način za predstavljanje korisničkog interfejsa.
- Korišćenje XAML-a ohrabruje razdvajanje korisničkog interfejsa od pozadinske logike.

- XAML je relativno jednostavan deklarativni jezik opšte namene koristan za konstruisanje i inicijalizaciju .NET objekata kao što su dugmad, checkbox, paneli, combobox i drugi.

C# je programski jezik koji se konstantno razvija, i koji je u poslednjih deset godina dostigao zavidan nivo. To je objektno-orijentisan programski jezik koji može da se primeni u razne svrhe, počev od klasičnih Windows aplikacija, preko Web aplikacija do aplikacija koje manipulišu bazama podataka. Dakle, spektar primena koji C# obuhvata je zaista širok. Druga pogodna osobina jeste to što je C# napravljen po ugledu na programske jezike C, C++ i Javu. Programeri koji imaju iskustva sa pomenutim jezicima za vrlo kratko vreme mogu da se osposobe za rad u C#-u. Projekat kreiran za potrebe ovog rada napisan je baš ovim jezikom.

Okruženje koje se uglavnom koristi u paru sa C# programskim jezikom je **Visual Studio**. Za svrhe ovog projekta korišćen je Visual Studio 2012. On u sebi sadrži pre svega editor koda koji je obogaćen dodatnim funkcionalnostima kao što su IntelliSense sistem, podrška za refaktorizaciju koda, itd. IntelliSense sistem predlaže imena unapred, čime olakšava korisnicima rad. Podrška za refaktorizaciju koda omogućava lakše uočavanje takozvanih *bad smell*-ova u kodu. *Bad smell* predstavlja loš deo koda, koji je neophodno ispraviti kako bi kod bio čistiji i pregledniji. Pored samog editora koda, Visual Studio 2012 ima i *debuger*.

Silverlight je Microsoftova platformska implementacija .NET Framework-a za veliki izbor pretraživača i operativnih sistema, koja omogućuje izradu, razvoj i predstavljanje obogaćenih internet aplikacija (*Rich Internet Applications* – RIA).

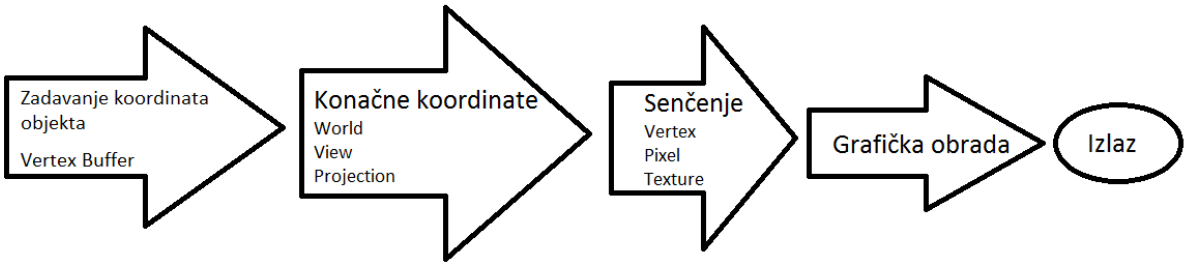
Silverlight omogućava rad sa vektorskom grafikom, tekstom, animacijom, interaktivnost, reprodukciju audio i video sadržaja, pristup podacima, kao i kontrole za multimedijalne i obogaćene internet aplikacije. U okviru Silverlight aplikacije, korisnički interfejs se deklariše u okviru XAML-a i programira koristeći podset .NET Framework-a.

Jedna od najvećih novina u Silverlight-5, koji je korišćen za potrebe ovog rada, je korišćenje 3D hardvera grafičkog adaptera za 3D vektorsku grafiku pomoću XNA framework-a. Kombinacija Silverlight i XNA omogućava potpuno novu klasu 3D aplikacija koje se pokreću direktno sa web-a. XNA framework daje dosta mogućnosti programerima da kontrolišu grafički hardver na prilično direktan način, a neki impresivni rezultati su mogući. Ali dok XNA znatno pojednostavljuje 3D programiranje, još uvek je relativno niskog nivoa tehnologije i Silverlight programeri bez prethodnog iskustva u 3D programiranju se mogu suočiti sa teškoćama na samom početku. S obzirom na to da je ovo nova tehnologija, dokumentacija je trenutno retka i nekoliko primera koji su dostupni, ne pokazuju uvek najlakši način da se nešto uradi.

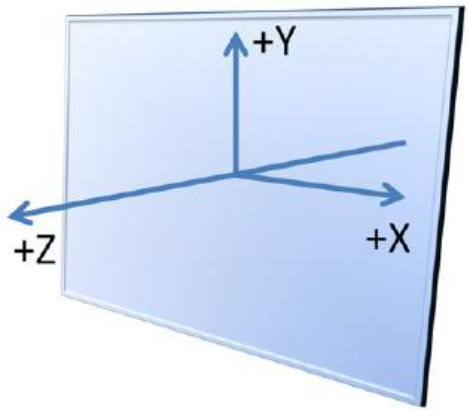
Za programiranje i razvoj Silverlight aplikacija može se koristiti Microsoft Visual Studio, standardni Microsoft-ov alat za razvoj aplikacija u .NET okruženju, kao i dodatak web pretraživaču za Silverlight (*plug-in*).

5. Kreiranje Silverlight 3D aplikacije

Kreiranje Silverlight 3D aplikacije može se posmatrati kao niz operacija (Slika 23). Prvo se zadaju koordinate tačaka objekta i način povezivanja istih. Zatim se vrše transformacije objekta, kojima se definiše položaj objekta na ekranu. Sledeći korak je senčenje, kojim se definiše kako se vide delovi objekta. Na kraju se izvršava grafička obrada i na taj način se dobija željeni izgled objekta na ekranu.



Slika 23. Niz operacija pri kreiranju Silverlight 3D aplikacije



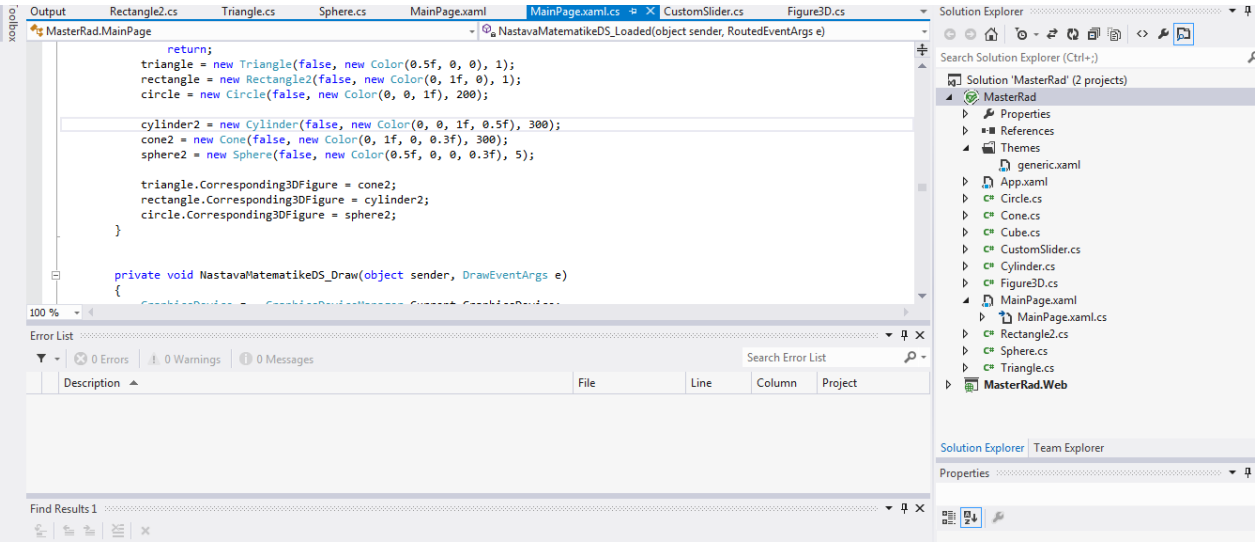
Slika 24. Položaj koordinatnog sistema u odnosu na ravan ekrana (xy ravan ekrana)

U narednom delu biće objašnjeni osnovni principi i klase koje se koriste za kreiranje Silverlight 3D aplikacije.

5.1. Početak

Nakon kreiranja Silverlight projekta, automatski su dodate dve XAML datoteke sa pozadinskim kodom: App.xaml (App.xaml.cs) i MainPage.xaml (MainPage.xaml.cs). App.xaml klasa predstavlja ulaznu tačku aplikacije i sadrži događaje *Startup*, *Exit* i *UnhandledException*.

U okviru metoda Application_Startup postavlja se početna stranica MainPage.xaml. MainPage klasa predstavlja Silverlight korisničku kontrolu (*user control*), tj. nasleđuje klasu UserControl. Dakle, sam izgled aplikacije se definiše u okviru MainPage.xaml datoteke, dok se obrada događaja definiše u MainPage.xaml.cs.



Slika 25. Izgled Visual Studio-a nakon kreirane Silverlight aplikacije

U HTML test page u elementu <object> se dodaje <param name="EnableGPUAcceleration" value="true" />. Ovo omogućava da se dobija ili postavlja vrednost koja ukazuje da li da se koristi GPU (engl. graphics processor unit) što potencijalno dovodi do grafičke optimizacije. Ovaj parametar se mora postaviti na pokretanju aplikacije. Ne može se promeniti nakon što Silverlight učita sadržaj.

Da bi se omogućilo iscrtavanje trodimenzionalne scene, potrebno je dodati event handler metod za događaj Draw površine za crtanje, tj. DrawingSurface.

U MainPage.cs treba dodati using Microsoft.Xna.Framework i using Microsoft.Xna.Framework.Graphics. Nepotrebne using direktive se moraju obrisati da ne bi došlo do različitih definicija koje su isto nazvane, poput System.Windows.Media (javile bi se dve definicije za Color).

5.2. Crtanje grafičkih primitiva (engl. Draw Primitive)

XNA framework je na prilično niskom nivou i radi direktno sa grafičkim hardverom. Jedine stvari koje zna kako da crta su **trouglovi i linije**. Kod koji ovo radi u programu napravljenom za potrebe ovog rada je

```
g.DrawPrimitives(PrimitiveType.TriangleList, 0, vb.VertexCount / 3);
g.DrawPrimitives(PrimitiveType.LineList, 0, vbMesh.VertexCount / 2);
```

Ovim se nalaže GPU-u da nacrta trouglove i linije pomoću zadatih temena.

5.3. Vertex Buffer

Sve tačke imaju 3D poziciju definisanu upotrebom **Vector3** strukture. Da bi program imao realno 3D osvetljenje, koristi se **VertexPositionNormalTexture** struktura**.** Temena trouglova predstavljaju se nizom. Niz temena se putem **VertexBuffer** klase šalje na GPU.

```
protected VertexBuffer vb;
VertexPositionNormalTexture[] vertices = CreateVertices(corners, facesArray);
vb = new VertexBuffer(g, VertexPositionNormalTexture.VertexDeclaration, vertices.Length, BufferUsage.WriteOnly);
vb.SetData(0,vertices,0,vertices.Length,VertexPositionNormalTexture.VertexDeclaration.VertexStride);
g.SetVertexBuffer(vb);
```

5.4. GraphicsDevice klasa

GraphicsDevice klasa se koristi za kontrolu GPU-a. Postoji samo jedna njena realizacija, koja se dobija kroz statičke osobine **GraphicsDeviceManager** klase. GraphicsDevice klasa ima metode za **crtanje osnova i za postavljanje vertex buffer-a** kako je već objašnjeno. Ona, takođe, ima dosta drugih svojstava i metode za kontrolu detalja u procesu 3D renderovanja. Osnovna podešavanja podrazumevaju samo crtanje prednjih strana trouglova, a ukoliko postoji potreba za crtanje i zadnjih strana `RasterizerState` se postavlja da bude `CullNone`.

```
GraphicsDevice g = GraphicsDeviceManager.Current.GraphicsDevice;
g.RasterizerState = RasterizerState.CullNone;
g.Clear(new Color(0.8f, 0.8f, 0.8f, 1.0f));
```

5.5. Efekti (engl. Effects)

Klasa **BasicEffect** je zapravo prilično moćna, ona obuhvata odgovarajuće (prirodno) 3D osvetljenje koje je postavljeno kao standardno, koje je i korišćeno u ovom radu. Pored ovog osvetljenja, postoje još dve vrste: Ambient koje osvetljava sve objekte i delove objekta podjednako i Directional koje osvetljava objekte u pravcu normale i to samo one delove koji su okrenuti ka izvoru svetlosti. Moguće je menjati i boju osvetljenja, kao i postavljati više različitih istovremenih osvetljenja.

```
be = new BasicEffect(g);
be.EnableDefaultLighting();
be.LightingEnabled = true;
```

5.6. Tekstura

U XNA, 3D scena se sastoji od mreža trouglova obloženih teksturama. Teksture su jednostavno 2D slike (engl. bitmap)-XNA klasa je **Texture2D**. Šalju se GraphicsDevice klasi koristeći Effect klasu i svakom temenu trougla dodeljuju se koordinate unutar svoje 2D teksture.

```
texture = new Texture2D(g, 1, 1, false, SurfaceFormat.Color);
texture.SetData<Color>(new Color[1] { color });
be.Texture = texture;
be.TextureEnabled = true;
```

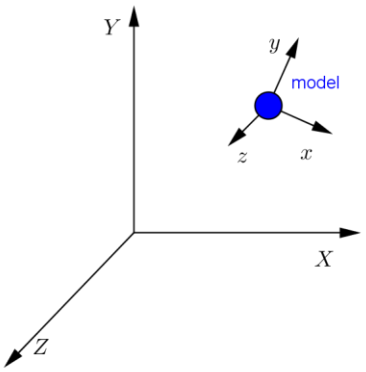
5.7. Matrice i 3D transformacije

Matrice i 3D transformacije su srce 3D programiranja. Tačke u VertexBuffer-u definišu mesto tačaka u 3D sistemu. Da bi se objekti prikazivali na ekrenu neophodno je prvo konvertovati 3D koordinate u 2D „ekranske" koordinate.

Postoje tri faze u ovom procesu, a svaka faza koristi 3D transformaciju definisanu pomoću **Matrix** klase. Matrix klasa zapravo predstavlja matricu 4×4 brojeva tipa float, ali njena osnovna funkcija je da transformiše 3D koordinate, definisane Vector3 klasom, kroz osnovne operacije translacija, rotacija i skaliranja (proporcionalno smanjivanje ili uvećavanje).

Prvi korak je konvertovanje koordinata modela na takozvane svetske (engl. world) koordinate. To se zove svetska transformacija (engl. world transform). Svetska transformacija se koristi da

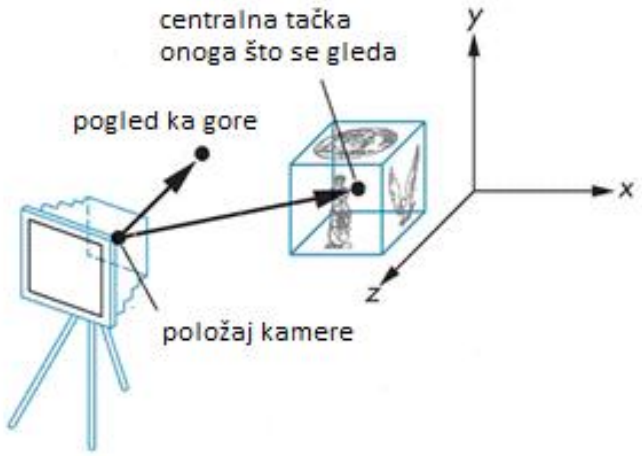
definiše gde je i kako je orijentisan model u virtuelnom 3D svetu. Ako 3D scena sadrži više modela, ili više delova istog modela, svaki će imati vlastite svetske transformacije.



Slika 26. *xyz* koordinatni sistem modela; *XYZ* koordinatni sistem sveta

```
Matrix rotate1 = Matrix.CreateFromQuaternion(Quaternion.CreateFromAxisAngle(axis,
(float)e.TotalTime.TotalSeconds * 3));
Matrix translate = Matrix.CreateTranslation(2, 0, (float)zoomValue);
Matrix rotate2 = Matrix.CreateRotationZ( MathHelper.TwoPi / 5 + (float)e.TotalTime.TotalSeconds / 3);
currentFigure.World = rotate1 * translate * rotate2;
```

Drugi korak je konvertovanje svetskih koordinata u koordinate pogleda, koje su orjentisane na tačku gledišta ili položaj kamere. To se zove transformacija pogleda (engl. view transform). Klasa Matrix ima pogodan metod (CreateLookAt) za stvaranje ove transformacije na osnovu položaja kamere i centralne tačke onoga što se gleda.

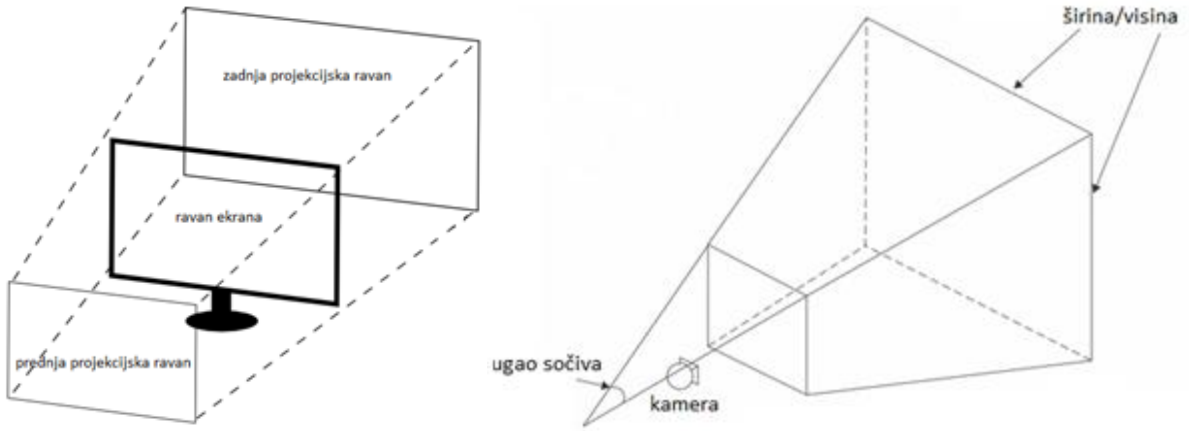


Slika 27. Postavljanje kamere

```
Matrix view = Matrix.CreateLookAt(new Vector3(0, 0, 8.0f), Vector3.Zero, Vector3.Up);
```

Prvo se zadaje položaj kamere, zatim centralna tačka onoga što se gleda i pogled na gore.

Treći korak je konvertovanje iz 3D koordinata u 2D koordinate ekrana. Radi doslednosti, ovaj korak se takođe vrši pomoću Matrix klase, pa je izlaz 3D koordinate, ali samo x i y vrednosti se koriste za kreiranje 2D slike na ekranu. Ovaj korak se naziva projekcijska transformacija (engl. projection transform) i XNA nudi dva izbora: orthogonal i perspective. Sa opcijom orthogonal, predmeti dalje od kamere ne izgledaju manji. Opcija perspective omogućava prirodniji izgled, udaljeniji predmeti izgledaju manji. Matrix klasa takođe pruža pogodan metod (CreatePerspectiveFieldOfView) za kreiranje projekcijske transformacije na osnovu osobine kamere, posebno ugla sočiva objektiva koji je izražen u radijanima na x osi.



Slika 28. Osobine kamere

```
Matrix projection = Matrix.CreatePerspectiveFieldOfView(0.85f, aspectRatio, 0.01f, 1000.0f);
```

Prvo se zadaje ugao sočiva, zatim odnos visine i širine projekcijske ravni, pa udaljenost prednje projekcijske ravni od kamere i na kraju udaljenost zadnje projekcijske ravni od kamere. Objekti koji treba da se vide se moraju nalaziti između projekcijskih ravni.

Nakon postavljanja ovih transformacija, BasicEffect klasa brine o postavljanju GraphicsDevice-a da renderuje 3D scenu na 2D ekran.

```
        be.World = world;
        be.View = view;
        be.Projection = projection;
```

5.8. Dodavanje efekata i tehnike (engl. Effect Passes and Techniques)

Tehnike obezbeđuju alternativne efekte renderovanja koji mogu da se izaberu u vreme crtanja (engl. draw-time), kao na primer u zavisnosti od vrste objekta koji se crta. BasicEffect klasa nudi jedinstvenu tehniku, koja je podrazumevano izabrana, a koja ima jedno dodavanje. To znači da u ovom radu svi kreirani objekti imaju iste efekte. Potrebno je omogućiti dodavanje efekata pre nego što se narede bilo kakve komande za crtanje.

```
be.CurrentTechnique.Passes[0].Apply();
```

5.9. Omogućavanje razmere DrawingSurface-a

Nakon kreiranja projekcijske transformacije, neophodno je omogućiti razmeru (širina/visina) DrawingSurface-a. Ovo se rešava kreiranjem lokalne promenljive aspectRatio i događaja (engl. event handler) u DrawingSurface-u SizeChanged u kome se promenljiva računa kao odnos širine i visine površi za crtanje. Ta promenljiva se zadaje kao odnos visine i širine projekcijske ravni pri definisanju projekcijske transformacije (5.7. str. 30).

```
private float aspectRatio = 1f;
private void triangulacijaDS_SizeChanged(object sender, SizeChangedEventArgs e)
        {
                aspectRatio = (float)(triangulacijaDS.ActualWidth /triangulacijaDS.ActualHeight);

        }
```

5.10. Invalidate

Na samom kraju Draw događaja poslednja linija koda je

```
e.InvalidateSurface();
```

Stavljanje ovog poziva na kraju Draw događaja dovodi do konstantnog učitavanja. Za 3D scenu koja se stalno menja, to omogućava glatke operacije. Ako je scena statična, ova linija koda je nepotrebna.

5.11. Bezbednosna ograničenja

Iz bezbednosnih razloga, korisnik mora eksplicitno da odobri web sajtu korišćenje ubrzane 3D grafike. Nažalost, ne postoji način da je programer odobri bez znanja korisnika.

```
private bool Is3dBlocked()
        {
            if (GraphicsDeviceManager.Current.RenderMode == RenderMode.Hardware)
                return false;
            string message;
            switch (GraphicsDeviceManager.Current.RenderModeReason)
            {
                case RenderModeReason.Not3DCapable:
                    message = "You graphics hardware is not capable of displaying this page ";
                    break;
                case RenderModeReason.GPUAccelerationDisabled:
                    message = "Hardware graphics acceleration has not been enabled on this web page.\n\n" +
                      "Please notify the web site owner.";
                    break;
                case RenderModeReason.TemporarilyUnavailable:
                    message = "Your graphics hardware is temporarily unavailable.\n\n" +
                      "Try reloading the web page or restarting your browser.";
                    break;
                case RenderModeReason.SecurityBlocked:
                    message =
                      "You need to configure your system to allow this web site to display 3D graphics:\n\n" +
                      "  1. Right-click the page\n" +
                      "  2. Select 'Silverlight'\n" +
                      "     (The 'Microsoft Silverlight Configuration' dialog will be displayed)\n" +
                      "  3. Select the 'Permissions' tab\n" +
                      "  4. Find this site in the list and change its 3D Graphics permission from 'Deny' to
                         'Allow'\n" +
                      "  5. Click 'OK'\n" +
                      "  6. Reload the page";
                    break;
                default:
                    message = "Unknown error";
                    break;
            }
            MessageBox.Show(message, "3D Content Blocked", MessageBoxButton.OK);
            return true;
        }
```

5.12. Optimizacija Draw događaja

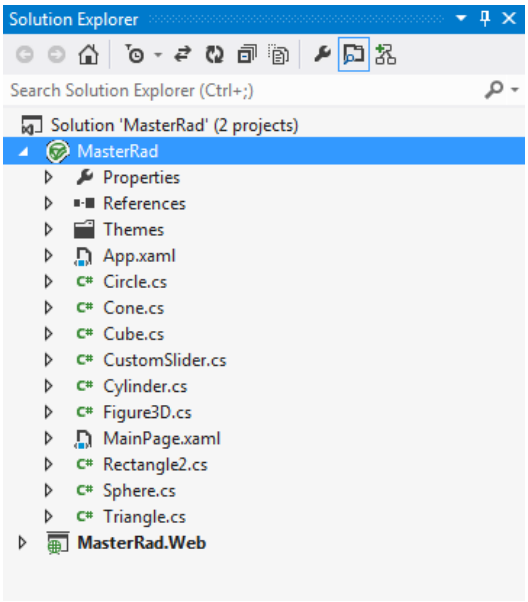
Sve animacije zasnivaju se na osobini ljudskog oka da registruje promenu u intervalu 2-3 stota dela sekunde. Iz tog razloga Draw događaj se izvršava konstantno, pa je poželjno da u njemu bude samo neophodan deo koda, a to je sve ono što utiče na stanje GraphicsDevice-a. Kreiranjem Loaded događaja u DrawingSurface-u i prebacivanjem svega iz Draw događaja što se izvršava samo jednom i ne utiče na stanje GraphicsDevice-a, ovaj problem se rešava, kao i kreiranjem promenljivih za klase Texture2D, Vertex Buffer i BasicEffect.

6. Klase koje predstavljaju 3D geometrijske objekte

Objektno orijentisani jezici pogodni su za implementaciju i prikaz grafičkih objekata, koji predstavljaju geometrijska tela.

Implementirana je apstraktna klasa <Figure3D>, koja predstavlja osnovu za sve geometrijske objekte i koja sadrži zajedničke osobine i metode.

Svaka vrsta objekata se predstavlja posebnom klasom u kojoj se definišu njegove osobine po kojima se razlikuje od ostalih objekata (na primer temena i stranice).



Slika 29. Klase koje čine projekat

Postupci generisanja valjka, kupe i sfere već je objašnjen u poglavlju 2.6. Ovde će biti predstavljene klase u programu kreiranom za potrebe ovog rada, deo u kome se definišu tačke i strane kojima se aproksimiraju ova tela, pri čemu se zna da je ravan ekrana u stvari xy ravan i pozitivni deo z ose je bliži kameri.

6.1. Valjak

Ovde će biti prikazana primena algoritma kojim se uzorkuje i aproksimira valjak. Valjak se aproksimira pravilnom n-tostranom prizmom. Kreira se valjak visine 1 i poluprečnika 1, čija osnova leži u xz ravni. Koristeći ovu klasu, moguće je konstruisati valjak različitih dimenzija. Menjanje visine se vrši množenjem y koordinate tačaka gornje baze, a poluprečnik osnove množenjem x i z koordinata svih uzorkovanih tačaka željenom veličinom. Položaj valjka na ravan ekrana se može menjati transformacijama i rotacijama, koristeći Matrix klasu kao što je već objašnjeno u poglavlju 4.7.

```
for (int i = 0; i < n; i++)
    {
        upperBase.Add(new Vector3((float)Math.Cos(i * 2 * Math.PI / n), 1, (float)Math.Sin(i * 2 * Math.PI / n)));
    }

for (int i = n-1; i >= 0; i--)
    {
        lowerBase.Add(new Vector3((float)Math.Cos(i * 2 * Math.PI / n), 0f, (float)Math.Sin(i * 2 * Math.PI / n)));
    }
            unitVertices.AddRange(upperBase);
            unitVertices.AddRange(lowerBase);
            int[] upperFace = new int[upperBase.Count];
            int[] lowerFace = new int[lowerBase.Count];
            int lowerBaseIndexPos = upperBase.Count;
            int vertexCount = unitVertices.Count;

            for (int i = 0; i < upperBase.Count; i++)
            {
                upperFace[i] = i;
                lowerFace[i] = i + lowerBaseIndexPos;
                int nextI = (i + 1) % lowerBaseIndexPos;
                faces.Add(new int[4] { i, vertexCount - i - 1, vertexCount - nextI - 1, nextI });
            }

            faces.Add(upperFace);
            faces.Add(lowerFace);
```

`unitVertices` predstavlja listu svih tačaka uzorkovanih iz valjka kojima se valjak aproksimira.

`faces` je lista nizova, svaki niz odgovara tačkama koje obrazuju jednu stranu, članovi niza su brojevi mesta tačke u `unitVertices` listi.

`upperBase` je lista tačaka gornje baze valjka, a `lowerBase` donje.

U prvoj for petlji se kreiraju tačke kojima se aproksimira gornja baza, analogno, u drugoj donja.

U `faces` se pored nizova koji predstavljaju baze dodaju i nizovi od četiri člana koji predstavljaju pravougaonike kojima se aproksimira omotač valjka.

Za $n = 3$ dobija se pravilna trostrana prizma, za $n = 4$ četvorostrana, a za dovoljno veliko n dobijamo valjak.

6.2. Kupa

Za namenu ovog projekta od važnosti je samo izgled tela, pa je klasa kojom se generiše kupa veoma jednostavna. Kupa se aproksimira pravilnom n-tostranom piramidom.

```
for (int i = 0; i < n; i++)
 {
    unitVertices.Add(new Vector3( (float)Math.Cos(i * 2 * Math.PI / n), 0,
     (float)Math.Sin(i * 2 * Math.PI / n)));
 }
            unitVertices.Add(new Vector3(0, 1, 0));
            int[] Base = new int[n];
            for (int i = 0; i < n; i++)
            {
                faces.Add(new int[3] { i, (i + 1) % n,n });
                Base[i] = i;
            }
            faces.Add(Base);
}
```

`unitVertices` je lista tačaka kojima se aproksimira kupa.

U for petlji se u listu prvo ubacuju tačke baze, a zatim se dodaje tačka vrha kupe.

Niz `Base` definiše mesta tačaka u `unitVertices` listi kojima se aproksimira baza kupe.

`faces` predstavlja listu nizova, svaki član niza je mesto tačke u `unitVertices` listi, ovde se povezuju tačke u trouglove i dodaje se baza na kraju.

Za dovoljno veliko n ovim se kreira kupa visine 1 i poluprečnika 1.

Za $n = 3$ dobija se pravilna trostrana piramida, $n = 4$ četvorostrana... Visina kupe se moze menjati množenjem y koordinate tačke vrha $(0,1,0)$, a poluprečnik osnove množenjem x i z koordinata tačaka baze sa željenom veličinom.

6.3. Sfera

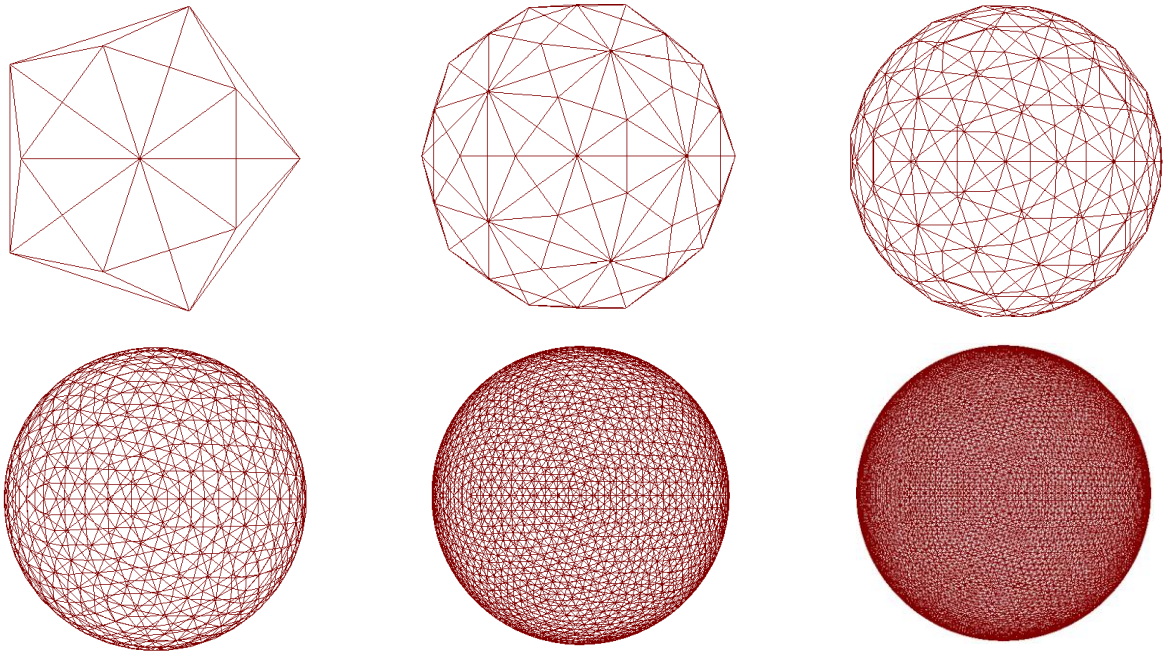
Kao što je i bilo reči, aproksimaciju sfere mrežom trouglova, gde je inicijalna triangulacija ikosaedar, optimalna je triangulacija sfere. U listu tačaka se prvo ubacuju temena ikosaedra kojih ima dvanaest, a zatim se oni povezuju u 20 trouglova. Bitno je voditi računa da se povežu odgovarajuća temena. Zatim se računa poluprečnik sfere kao kvadratni koren iz zbira kvadrata koordinata proizvoljne tačke iz liste tačaka. Zatim ikosaedar ulazi u drugu for petlju gde se računaju središnje tačke svake ivice svakog trougla (`midPointA, midPointB, midPointC`).

Te tačke se projektuju na sferu tako što se svaka koordinata tačke množi odgovarajućim parametrom (`tA, tB, tC`). Nove tačke (`midPointAtoSphere, midPointBtoSphere, midPointCtoSphere`) dodaju se u listu tačaka i od svakog trougla se prave četiri nova od temena trougla i novodobijenih tačaka i dodaju u pomoćnu praznu listu `faces2`.

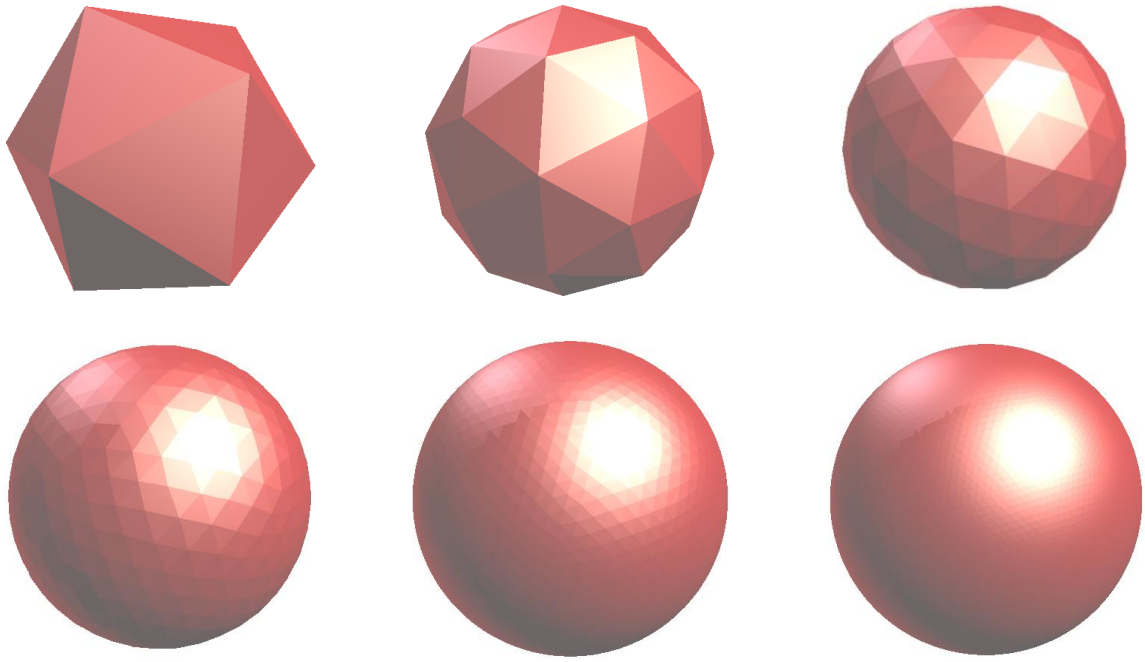
Lista `faces` se čisti, iz nje se izbacuju sve stranice ikosaedra i dodaju nove, odnosno `faces2,` i dobija se novo telo koje predstavlja bolju aproksimaciju sfere. Za novodobijeno telo proces je analogan. Ovaj proces se ponavlja *n* puta (prva for petlja).

```
unitVertices = new List<Vector3>(); faces = new List<int[]>(); List<int[]> faces2 = new List<int[]>();
unitVertices.Add(new Vector3( 0, 0, 1f )); unitVertices.Add(new Vector3( 0.894f, 0, 0.447f ) );
unitVertices.Add(new Vector3(0.276f,0.851f,0.447f)); unitVertices.Add(new Vector3(-0.724f,0.526f, 0.447f));
unitVertices.Add(new Vector3(-0.724f,-0.526f,0.447f)); unitVertices.Add(new Vector3(0.276f,-0.851f,0.447f));
unitVertices.Add(new Vector3(0.724f,0.526f,-0.447f)); unitVertices.Add(new Vector3(-0.276f,0.851f,-0.447f));
unitVertices.Add(new Vector3( -0.894f, 0.000f,-0.447f));unitVertices.Add(new Vector3(-0.276f,-0.851f,-0.447f));
unitVertices.Add(new Vector3( 0.724f,-0.526f,-0.447f  ));unitVertices.Add(new Vector3(0, 0, -1f));
        faces.Add(new int[3] {0,2,1}); faces.Add(new int[3]{0,3,2});
        faces.Add(new int[3] {0,4,3});faces.Add(new int[3] {0,5,4});
        faces.Add(new int[3] {0,1,5});faces.Add(new int[3] {11,6,7});
        faces.Add(new int[3] {11,7,8});faces.Add(new int[3] {11,8,9});
        faces.Add(new int[3] {11,9,10});faces.Add(new int[3] {11,10,6});
        faces.Add(new int[3] {1,2,6});faces.Add(new int[3] {2,3,7});
        faces.Add(new int[3] {3,4,8});faces.Add(new int[3] {4,5,9});
        faces.Add(new int[3] {5,1,10});faces.Add(new int[3] {6,2,7});
        faces.Add(new int[3] {7,3,8});faces.Add(new int[3] {8,4,9});
        faces.Add(new int[3] {9,5,10});faces.Add(new int[3] {10,1,6});
        float r = (float) Math.Sqrt(unitVertices[0].X*unitVertices[0].X+unitVertices[0].Y*unitVertices[0].Y+
        unitVertices[0].Z*unitVertices[0].Z);
        for (int i = 0; i <approximationQuality; i++)
        {
            faces2.Clear();
            for (int j = 0; j < faces.Count; j++)
            {
                Vector3 midPointA = (unitVertices[faces[j][0]] + unitVertices[faces[j][1]]) / 2;
                Vector3 midPointB = ((unitVertices[faces[j][1]] + unitVertices[faces[j][2]]) / 2);
                Vector3 midPointC = ((unitVertices[faces[j][2]] + unitVertices[faces[j][0]]) / 2);
                double tA = r / (Math.Sqrt((midPointA.X) * (midPointA.X) + (midPointA.Y) * (midPointA.Y)
                 + (midPointA.Z) * (midPointA.Z)));
                double tB = r / (Math.Sqrt((midPointB.X) * (midPointB.X) + (midPointB.Y) * (midPointB.Y)
                 + (midPointB.Z) * (midPointB.Z)));
                double tC = r / (Math.Sqrt((midPointC.X) * (midPointC.X) + (midPointC.Y) * (midPointC.Y)
                 + (midPointC.Z) * (midPointC.Z)));
                Vector3 midPointAtoSphere = new Vector3(midPointA.X * (float)tA, midPointA.Y * (float)tA,
                 midPointA.Z * (float)tA);
                Vector3 midPointBtoSphere = new Vector3(midPointB.X * (float)tB, midPointB.Y * (float)tB,
                 midPointB.Z * (float)tB);
                Vector3 midPointCtoSphere = new Vector3(midPointC.X * (float)tC, midPointC.Y * (float)tC,
                 midPointC.Z * (float)tC);
                unitVertices.Add(midPointAtoSphere); unitVertices.Add(midPointBtoSphere);
                unitVertices.Add(midPointCtoSphere);
                faces2.Add(new int[3] { faces[j][0], unitVertices.Count - 3, unitVertices.Count - 1 });
                faces2.Add(new int[3] { faces[j][1], unitVertices.Count - 2, unitVertices.Count - 3 });
                faces2.Add(new int[3] { faces[j][2], unitVertices.Count - 1, unitVertices.Count - 2 });
                faces2.Add(new int[3] { unitVertices.Count - 3, unitVertices.Count - 2,
                 unitVertices.Count - 1 });
            }
            faces.Clear();
            faces.AddRange(faces2);
```

Datim kodom je generisana jedinična sfera. Promena poluprečnika vrši se množenjem svih koordinata svih tačaka u listi `unitVertices` željenom veličinom.



Slika 30. Od ikosaedra do sfere (mreža trouglova)



Slika 31. Od ikosaedra do sfere (sa teksturom)

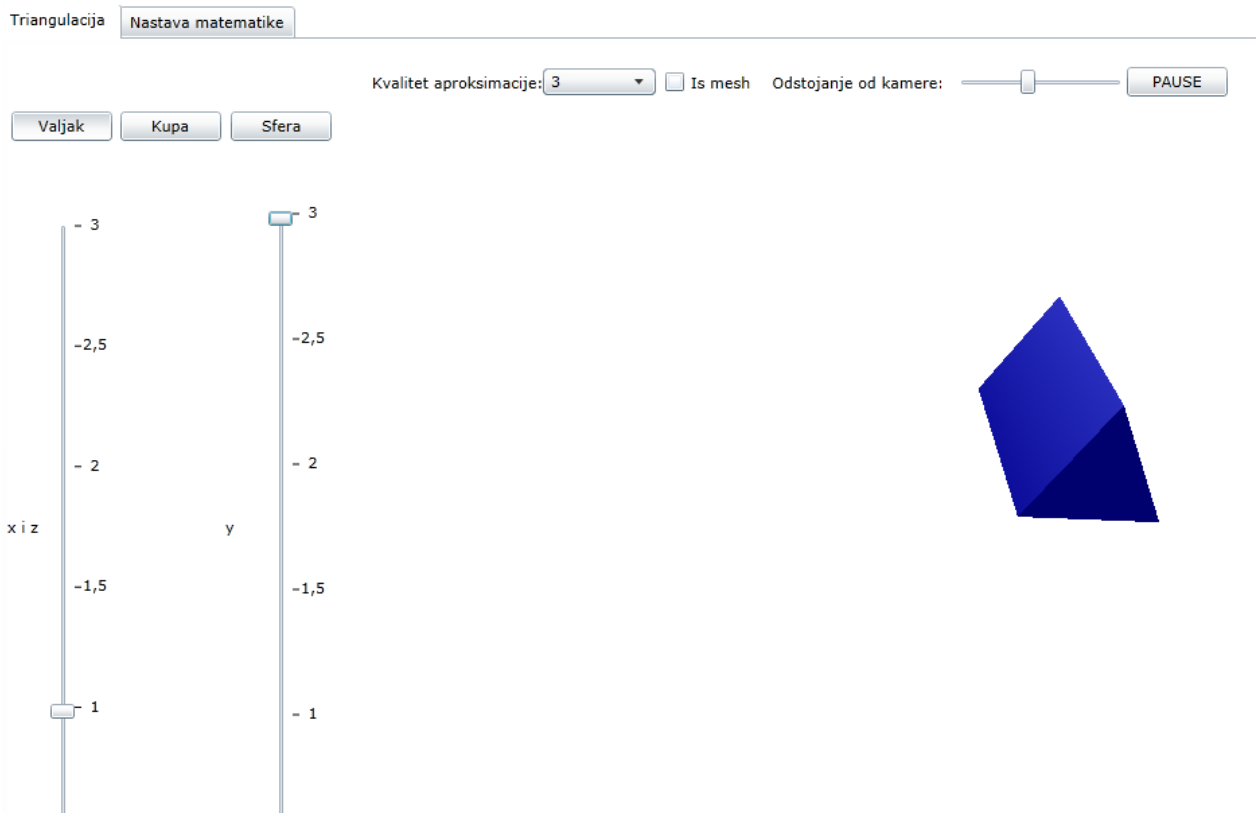
7. Silverlight 3D aplikacije

Na osnovu ranije opisanih teorijskih osnova i programerskih tehnika, razvijena je Silverlight 3D aplikacija koja omogućava :

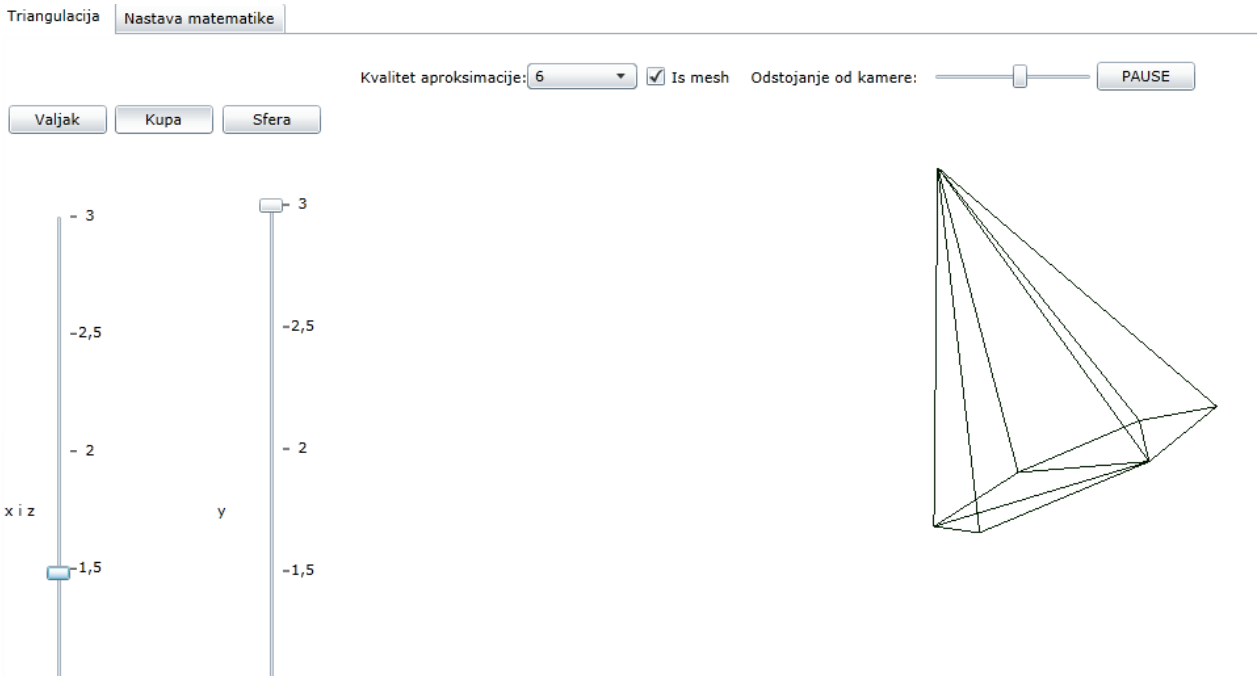
- prikaz triangulacije i aproksimacije tela
- transformacije tela
- promene izgleda tela pri promeni parametara
- primenu aplikacije u nastavi matematike

U ovom delu dajemo pregled razvijene aplikacije.

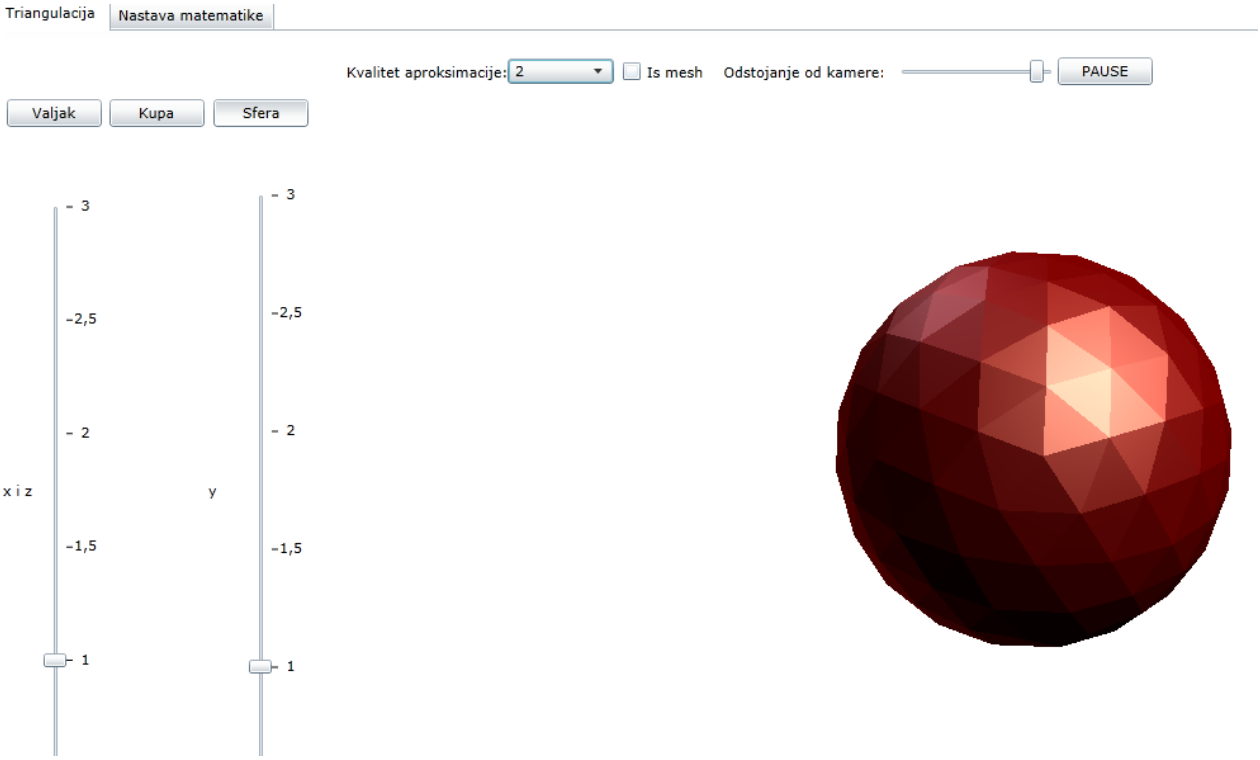
Prvi deo ove aplikacije osmišljen je tako da prikazuje način aproksimacije kreiranih objekata. Iz tog razloga, postoji mogućnost izbora kvaliteta aproksimacije, kao i prikazivanje objekata putem mreže trouglova ili sa teksturom. Ovaj deo napravljen je u cilju razumevanja načina aproksimacije. Odabirom kvaliteta aproksimacije se, u stvari, bira do kog broja se „ide“ u for petljama u programu prikazanom u poglavlju 6. Omogućena je promena parametara tela čime se tela dinamički menjaju na ekranu.



Slika 32. Aplikacija prikazuje trostranu prizmu koja je aproksimacija valjka

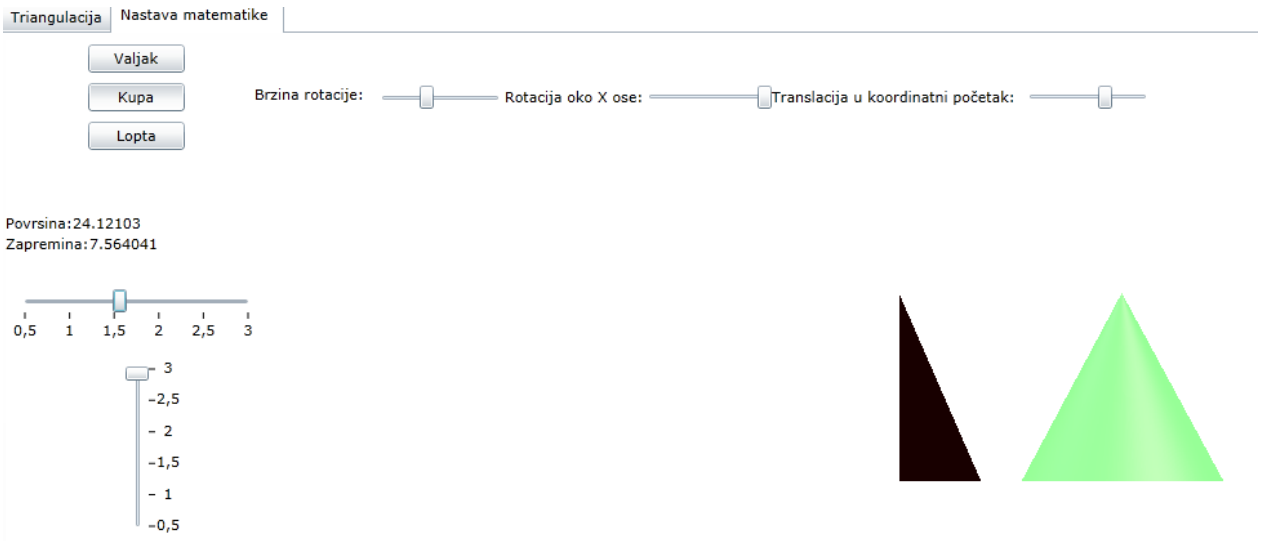


Slika 33. Aplikacija prikazuje mrežu trouglova koja određuje šestostranu piramidu



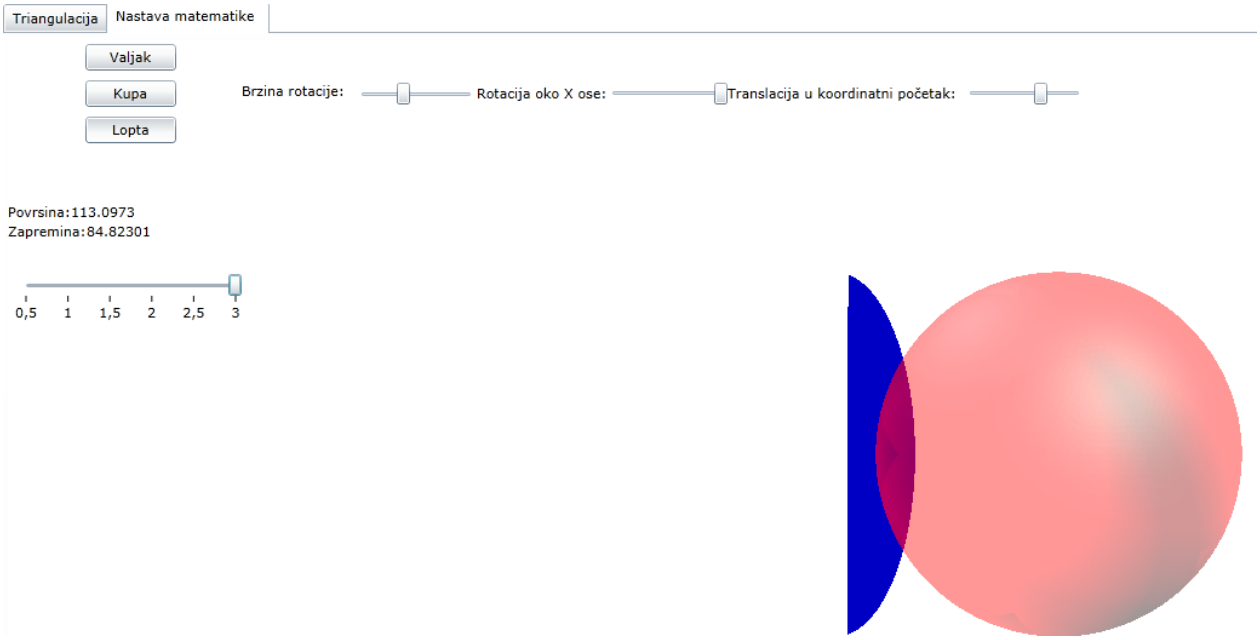
Slika 34. Aplikacija prikazuje telo koje predstavlja aproksimaciju sfere

Drugi deo ove aplikacije napravljen je kao primer primene u nastavi matematike. Upotrebom ovog edukativnog softvera koji koristi 3D geometriju, nastavnik bi lakše objasnio koja tela se dobijaju rotacijom pravougaonika, trougla i polovine kruga. Omogućena je promena brzine rotacije figura, kao i promena ugla gledanja. Slajderom se vrši i translacija figure i tela u koordinatni početak, čime se uočava da telo u potpunosti prekriva rotirajuću figuru. Postoje i slajderi kojima se mogu menjati dimenzije figura i tela. Iznad slajdera prikazuje se vrednost zapremine i površine posmatranih tela, naravno, ove vrednosti se menjaju pri promeni dimenzija tela.



Slika 35. Aplikacija prikazuje rotirajući trougao i kupu koja nastaje rotacijom trougla

Slika 36. Aplikacija prikazuje rotirajući pravougaonik i valjak koji nastaje rotacijom pravougaonika



Slika 37. Aplikacija prikazuje rotirajući polukrug i loptu koja nastaje rotacijom polukruga

Korišćenjem ove aplikacije nastava će učenicima biti prihvatljivija, razumljivija, a samim tim što je nešto novo i drugačije, biće im sigurno i interesantija. Ukoliko se neko od učenika zainteresuje više, moguće je i na sekciji matematike praviti aplikacije koje će se koristiti u nastavi.

U vreme kada gotovo svaka kuća ima kompjuter i pristup internetu i kada deca bolje i brže rade na računarima i mobilnim uređajima od svojih roditelja i veliki deo slobodnog vremena provode na istom, kada su čitanje knjiga zamenili fejsbukom, igranje loptom zamenili igricama na računaru, zašto im onda ne bismo omogućili i pristup nastavnim materijalima putem interneta? Načini za ovo su mnogobrojni: pravljenje blogova od strane nastavnika, grupa na fejsbuk društvenoj mreži i možda najzahtevnije, pravljenje web sajta. Za postavljanje kreirane aplikacije u korišćenoj tehnologiji na web sajt koriste se fajlovi TestPage.html, Silverlight.js i ClientBin.

Postavljanjem kreirane aplikacije na web sajt, učenicima je omogućeno da u bilo koje vreme i na bilo kom mestu gde imaju pristup internetu samostalno uočavaju matematičke zakonitosti, eksperimentišu i uče na kreativan, samostalan i njima zanimljiviji način.

8. Zaključak

Matematičke osnove aproksimacije datih tela mrežom trouglova, kao i transformacije tela primenjena u projektu je jasno predstavljena u prvom delu rada, bez njenog shvatanja nemoguće je kreirati željene objekte.

Velika i glavna prednost kreirane biblioteke jeste ta što su moguća dodavanja novih objekata na jednostavan način, dodavanjem nove klase u kojoj bi se definisala samo lista temena objekta i lista nizova koji obrazuju stranice objekta, čak nije neophodno ni razumevanje programa u celosti, dovoljno je koristiti analogiju pri kreiranju novih 3D tela.

Što se tiče primene u nastavi matematike, dat je samo primer, ali mogućnosti nadograđivanja i pravljenja sličnih aplikacija zavisi samo od želje i mašte pojedinca. Ja ću sigurno u budućnosti raditi na kreiranju novih, jer se na ovaj način budi veće interesovanje kod učenika i približava im se gradivo matematike.

Ovaj rad je napisan na jednostavan način, kao uputstvo za kreiranje Silverlight 3D aplikacije uz detaljna objašnjenja za one bez iskustva u 3D programiranju. Razmatrana su samo znanja potrebna za kreiranje aplikacije koja prikazuje 3D geometrijska tela koja se može primenjivati u nastavi matematike, što je i bio cilj. Zainteresovani mogu pogledati aplikaciju napravljenu za potrebe ovog rada i koristiti je u razne svrhe, možda i poboljšati. Za potpuno razumevanje rada neophodno je koristiti i rad i dostupnu aplikaciju na vesnarvovic.wordpress.com na SkyDrive-u.

Literatura:

[1] Nadrljanski Đ., *Obrazovni softver –Hipermedijalni sistemi*, Univerzitet u Novom Sadu, Tehnički fakultet „Mihajlo Pupin“ u Zrenjaninu, Zrenjanin, 2000, str. 106.

[2] Radosav D., *Obrazovni računarski softver i autorski sistemi*, Univerzitet u Novom Sadu, Tehnički fakultet „Mihajlo Pupin“ u Zrenjaninu, Zrenjanin, 2005, str. 10.

[3] G. D. Glejzer, *Geometrija u školi*, Prvi septembar 34, 1995, str. 3-4

[4] Dubravka Glasnović Gracin, *Računalo u nastavi matematike: Teorijska podloga i metodičke smjernice (1. dio: Potencijali primjene računala u nastavi)*, Matematika i škola 46 (2008); 10-15

[5] E. N. Wiebe, *The Taxonomy of Geometry and Graphics*, JGG, Vol. 2, (1998) 189-196.

[6] Lester, J., *Designing interactive mathematics*
http://oldweb.cecm.sfu.ca/~jalester/DesignIntMath.pdf

[7] Vladimir Poljak, *Didaktika*, Školska knjiga 1984.

[8] Robert Bowen and Stephen Fisk, *Generation of Triangulations of the Sphere*, *Mathematics of Computation*, Vol. 21, No. 98 (1967), pp. 250-252

[9] Mea Bombardelli, *Eulerova formula*, Matematika i škola 19 ,2003, str. 179-182

[10] O'Rourke, J. §2.3 in *Computational Geometry in C, 2nd ed.* Cambridge, England: Cambridge University Press, 1998.

[11] Wickham-Jones, T. *Mathematica Graphics: Techniques and Applications.* New York: Springer-Verlag, pp. 406 and 448, 1994.

[12] Amato, N. M.; Goodrich, M. T.; and Ramos, E. A. *A Randomized Algorithm for Triangulating a Simple Polygon in Linear Time. Discr. Comput. Geom.* **26**, 245-265, 2001.

[13] Jovan Kečkić, *Matematika 3 sa zbirkom zadataka za opštu gimnaziju i gimnaziju prirodno matematičkog smera*, Zavod za udžbenike i nastavna sredstva, Beograd

[14] Pete Brown, *Silverlight in action*, Manning Publications Co, 2010, | ISBN-13: 978-1935182375

[15] Olga Bodroža-Pantić, *Kombinatorna geometrija*, Novi Sad: Univerzitet u Novom Sadu-PMF, 2000

[16] Guoliang Xu, *Discrete Laplace-Beltrami Operator on Sphere and Optimal Spherical Triangulations*, The Institute of Computational Mathematics, Chinese Academy of Sciences, Beijing, China

[17] Wolfang Kühnel; trensleted by Bruce Hunt, *Differential Geometry: Curves-Surfaces-Manifolds*, Second Edition, American Mathematical Society, 2006.

[18] Dragan Mašulović, *Odabrane teme diskretne matematike*, Novi Sad: Prirodno-matematički fakultet, 2007.

[19] Krunislav Mihailović, Krsta Vračarić, *Geodetska merenja i računanja (praktikum): za II, III i IV razred geodetske škole*, Zavod za udžbenike i nastavna sredstva, Beograd

[20] И. И. Олвховский, *Курс теоретической механики дляфизиков,* Издателвство Московского универзитета, 1974.

[21] http://www.wolfram.com/products/webmathematica/examples/

[22] http://www.wolframalpha.com/examples/

[23]http://marul.ffst.hr/odsjeci/uciteljski/nastava/metodika%20nastave%20matematike/MNM%201%202013_2014/NASTAVNA%20SREDSTVA%20I%20POMAGALA.pdf

[24] www.geogebra.org

[25] www.wikipedia.org

[26] http://inside.mines.edu/fs_home/gmurray/ArbitraryAxisRotation/

[27] http://www.csee.umbc.edu/~squire/reference/polyhedra.shtml#icosahedron

Biografija

Vesna Rvović rođena je 11.3.1989. u Kikindi. Osnovnu školu „Žarko Zrenjanin“ završila je u Kikindi sa prosečnom ocenom 5. Nakon osnovnog obrazovanja upisala je Gimnaziju „Dušan Vasiljev“ u Kikindi, prirodno-matematički smer. U toku školovanja u svom rodnom gradu učestvovala je na brojnim takmičenja iz matematike, fizike i iz srpskog jezika i osvajala prva mesta na opštinskim i okružnim takmičenjima, a iz matematike stizala je i do republičkih takmičenja. Od svoje osme godine trenirala je košarku, sve dok nije napustila Kikindu kako bi nastavila školovanje na Prirodno-matematičkom fakultetu u Novom Sadu 2008. godine. Osnovne studije u trajanju od tri godine na smeru primenjena matematika (matematika finansija) završila je u roku sa prosečnom ocenom 8,32 i osvojenim 190 ESPB bodova. U junu 2011. je stekla zvanje matematičar-primenjena matematika, a u septembru položila razliku ispita i upisala akademske-master studije na smeru matematika (profesor matematike) u trajanju od dve godine na istom fakultetu. Sve ispite predviđene studijskim programom položila je do juna 2013. godine sa prosečnom ocenom 8,76. Od septembra do decembra 2013. radila je u Osnovnoj školi „Sveti Kirilo i Metodije“ u Novom Sadu kao nastavnik matematike. Od 15.1.2014. zaposlena je u Gimnaziji „Dušan Vasiljev“ u Kikindi kao profesor matematike.

Ključna dokumentacijska informacija

Redni broj: RBR	
Identifikacioni broj: IBR	
Tip dokumentacije: TD	Monografska dokumentacija
Tip zapisa: TZ	Tekstualni štampani materijal
Vrsta rada: VR	Master rad
Autor: AU	Vesna Rvović
Mentor: MN	Prof. dr Đorđe Herceg
Naslov rada: NR	Primena Silverlight-a za prikazivanje geometrijskih tela
Jezik publikacije: JP	Srpski (Latinica)
Jezik izvoda: JI	Srpski
Zemlja publikovanja: ZP	Srbija
Uže geografsko područje: UGP	Vojvodina
Godina: GO	2014.
Izdavač: IZ	Autorski reprint
Mesto i adrsa: MA	
Fizički opis rada: FO	
Naučna oblast: NO	
Naučna disciplina: ND	
Predmetne odrednice, Ključne reči: PO	
UDK	
Čuva se: ČU	Biblioteka departmana za matematiku i informatiku
Važna napomena: VN	
Izvod: IZ	

Datum prihvatanja teme od strane NN veća: DP		
Datum odbrane: DO		
Članovi komisije: KO	Predsednik:	
	Mentor:	Dr Đorđe Herceg, redovan profesor Prirodno-matematičkog fakulteta u Novom Sadu
	Članovi komisije:	