SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 83

ALAT ZA PRIANJANJE PROTEINA: MODUL ZA VIZUALIZACIJU

Ivan Sović

Zagreb, lipanj 2010.

Hvala mami i tati na svemu, sestri Ani što je uvijek bila tu kada bi zapelo i Mili Šikiću na velikom strpljenju i svim pruženim prilikama.

Sadržaj

1. Uvod

Prianjanje proteina je postupak određivanja strukture kompleksa nastalih spajanjem (ili prianjanjem) dvaju proteina. Ovim postupkom modelira se jedna od važnijih karakteristika biokemijskih reakcija, pri čemu se jedan od proteina (manji) naziva ligand, a drugi (veći) receptor. Većina metoda uključuje ispitivanje svih mogućih konformacija[1] tih dvaju proteina, te uporabom određenog kriterija razlučivanje koja od njih najbolje odgovara kristalografskoj (stvarnoj) vrijednosti. Ako su kvalitetno izvedeni, ovakvi sustavi mogu u praksi značajno ubrzati i poboljšati postupak traženja novih lijekova.

Protein Docking Tool (PDT) je postojeći sustav koji izvršava upravo opisano prianjanje proteina. On implementira metodu razvoja površine proteina u red kuglinih funkcija. Ova metoda se još naziva i SPF metoda, što je skraćeno od eng. *Spherical Polar Fourier.* Sam postupak pretvorbe površine proteina u koeficijente reda kuglinih funkcija složen je postupak s nekoliko međukoraka. Kako svaki od ovih koraka zapravo opisuje trodimenzionalnu strukturu proteina, to omogućuje izradu alata koji će pružiti uvid u njezin izgled. Sam alat trebao bi moći vizualizirati izgled proteina prije ulaska u proces prianjanja, izgled površine nakon razvoja u red kuglinih funkcija, te prikazati komplekse dobivene kao rezultate procesa prianjanja proteina. Osim toga, korisnik bi trebao biti u mogućnosti interaktivno pomicati i rotirati učitane proteine po ekranu, te pokretati proces prianjanja iz alata. Razvoj takvog alata upravo je cilj ovog rada.

U 2. poglavlju iznesena je teorija potrebna za razumijevanje pojedinih dijelova rada. Općeniti opis sustava, kao i prikaz njegove strukture dani su u 3. poglavlju. Detaljniji opis alata i pojedinih njegovih dijelova iznesen je u 4. poglavlju. 5. poglavlje sadrži prikaz testiranja alata, dok su u 6. poglavlju dani naputci za njegovo pokretanje i korištenje. Na kraju je iznesen zaključak, te dan kratak sažetak na hrvatskom i engleskom jeziku, kao i popis literature, u poglavljima 7, 8, 9 i 10.

[1] Relativnih položaja u prostoru pretraživanja.

2. Teorijska pozadina

Ovo poglavlje obuhvaća opis različitih načina prikaza molekula, načine bojanja atoma, metodu prikaza trodimenzionalnih modela, postupak prianjanja proteina na način kako ga implementira sustav PDT, algoritam za rekonstrukciju površine molekule iz zadanog grida te novi postupak ocjenjivanja točnosti rezultata prianjanja proteina. Svaka od ovih cjelina implementirana je u alat razvijan u sklopu ovog rada, a njihova teoretska osnova nužna je za razumijevanje rada alata.

2.1. Načini prikaza molekula

Postoji više metoda prikaza molekula. Neki standardni načini su: kalotni, žičani, prikaz štapićima, prikaz kuglama i štapićima, te prikaz pomoću površine molekule. Također, za potrebe uvida u rezultate procesa prianjanja proteina, nužna je i mogućnost prikaza površine zadane molekule dobivene kuglinim funkcijama te raspoređivanjem na grid [1]. U nastavku slijedi pojašnjenje svakog od ovih načina prikaza.

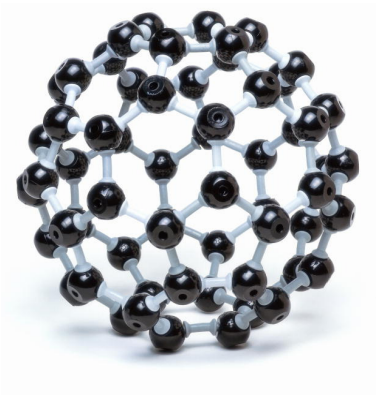
2.1.1. Prikaz kuglama i štapićima

Prikaz kuglama i štapićima sastoji se od atoma sadržanih u molekuli, te veza među njima. Atomi su prikazani kuglama, te su povezani ravnim linijama koje spajaju njihova središta. Linije predstavljaju kovalentne veze među tim atomima. Dvostruke i trostruke veze često se prikazuju zavojnicama koje tvore zaobljene poveznice između kugli.

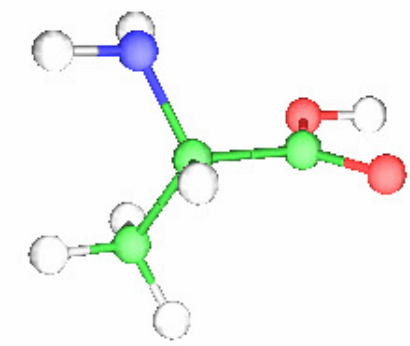
Kovalentna veza je vrsta kemijskog povezivanja koje karakterizira dijeljenje para elektrona između atoma. Drugim riječima, to je stabilnost između privlačenja i odbijanja atoma koja nastaje kada oni dijele elektrone.

U stvarnosti se linije koje predstavljaju veze atoma izvode pomoću žica ili krutih štapića (eng. *sticks*), dok se atomi predstavljaju kuglama (eng. *balls*), te otuda i dolazi ime ovog modela prikaza. Kod računalnog prikaza, za iscrtavanje kugli i štapića upotrebljavaju se odgovarajući prostorni modeli, čime se postiže realističnost prikaza i

sličnost sa fizičkim izgledom modela molekule. Slike 1 i 2 prikazuju modele molekula predstavljene kuglama i štapićima.



Slika 1. Izgled modela prikazanog kuglama i štapićima.

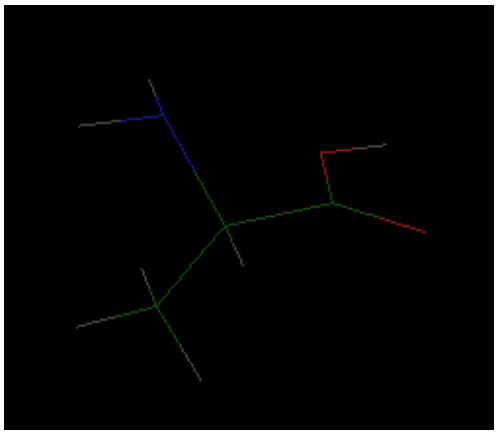


Slika 2. Prikaz aminokiseline Alanin pomoću kugli i štapića.

Kod ove vrste prikaza molekula kutovi među vezama, te njihove duljine, odgovaraju stvarnim odnosima unutar molekule, dok prostor kojeg atomi zauzimaju nije uopće prikazan, ili je predstavljen u nekom osnovnom obliku s relativnim veličinama kugli. Prednost prikaza pomoću kugli i štapića jest bolji uvid u strukturu povezanosti unutar molekule, za razliku od modela molekula koji popunjavaju prostor.

2.1.2. Žičani prikaz i prikaz štapićima

Ova vrsta prikaza zapravo je pojednostavljeni prikaz pomoću kugli i štapića. Žičani model sastoji se samo od veza između povezanih atoma. One su povučene iz središta svakog atoma molekule. Ova vrsta modela ne sadrži kugle kojima su predstavljeni atomi, te također korisniku ne daje nikakve informacije o veličini pojedinih atoma. Dijelovi veza među atomima mogu biti predstavljeni raznim bojama, čime se povećava preglednost modela, te daje korisniku informacija o atomima čija su središta povezana. Razlika između žičanog prikaza i prikaza štapićima je što se kod prikaza štapićima umjesto linija poveznica atoma koriste trodimenzionalni modeli koji povezuju središta tih atoma (kao i kod prikaza kuglama i štapićima).
Slika 3 prikazuje žičani model molekule sa slike 2.

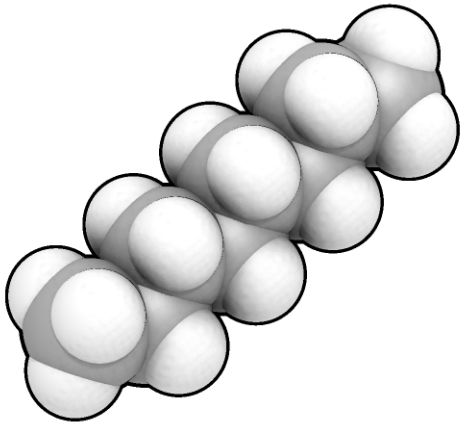


Slika 3. Žičani prikaz molekule

2.1.3. Kalotni prikaz

Kalotni modeli (eng. *Calotte models*) pripadaju skupini trodimenzionalnih modela koji popunjavaju prostor. Oni zapravo predstavljaju daljnji logični razvoj modela sa kuglama i štapićima. Koriste se za detaljniji prikaz molekula. Atomi pojedinih molekula prikazani su kuglama raznih boja. Također, kugle su različitih radijusa, koji je ovisan o kemijskom elementu atoma. Veličina atoma, njihovi položaji, te veze među elementima odgovaraju njihovim odnosima u stvarnosti.

Slika 4 predstavlja kalotni prikaz molekule oktana. Ona se sastoji od 18 atoma vodika (prikazanih bijelom bojom), te 8 atoma ugljika (prikazanih sivom bojom).

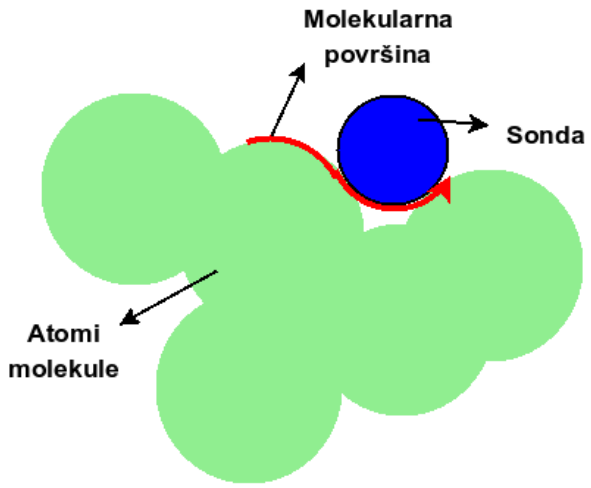


Slika 4. Kalotni prikaz jednog izomera oktana.

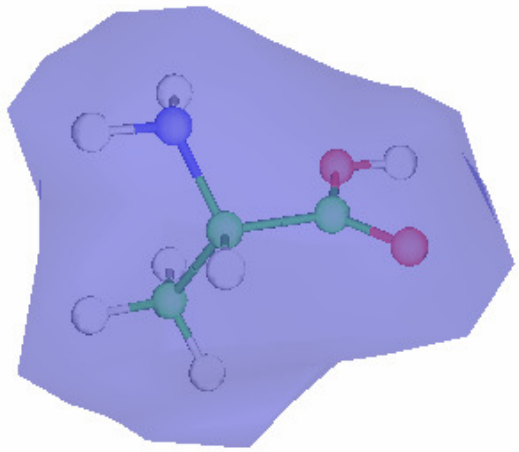
2.1.4. Molekularna površina

Površina molekule je vanjska ovojnica te molekule, tj. prostor koji bi bio dostupan nekoj tekućini (eng. *solvent*) izvan nje [8]. Kao tekućina najčešće se uzima molekula vode. Prilikom određivanja površine, molekula tekućine reprezentirana je kuglom koja se naziva *sonda* (eng. *probe*), a čiji radijus (za vodu) je jednak 1.4Å. Slika 5 prikazuje postupak određivanja molekularne površine. Kako bismo stvorili ovakav model prikaza, potrebne su nam informacije o položajima svih atoma unutar molekula, te njihovi radijusi. Prikaz površine aminokiseline alanin superponiran iznad modela kugli i štapića dan je na slici 6.

Prikaz površine molekule potreban je i iznimno praktičan primjerice kod alata čija je svrha prianjanje proteina. On omogućuje korisniku jednostavno i brzo provjeravanje točnosti ponuđenog rješenja. Nedostatci ovakvog načina prikaza molekule su otežano raspoznavanje pojedinih atoma, te nedostatak informacija o vezama među njima.



Slika 5. Postupak određivanja molekularne površine.



Slika 6. Prikaz površine aminokiseline Alanin.

2.2. Bojanje atoma

Za prikaz pojedine molekule korištene su razne boje kako bi se predočila razlika između atoma, aminokiselina, lanaca ili nekih drugih svojstava. Pojedini atomi u praksi najčešće su bojeni po *CPK* sustavu boja (naziv je skraćenica imena autora: Corey, Pauling i Kutlin) [4]. CPK sustav definira boje za svaki poznati atom. Tablica 1 izlaže pregled boja za atome koji se najčešće nalaze u molekulama proteina.

Tablica 1. CPK sustav boje atoma.

Atom	Boja
Dušik	plava
Kisik	crvena
Sumpor	žuta
Ugljik	siva, crna, zelena
Vodik	bijela
Natrij	tamnoplava
Fosfor, željezo, barij	tamnožuta
Flor, silicij, zlato	svjetložuta
Magnezij	tamnozelena
Nikal, bakar, cink, bronca, litij	smeđa
Aluminij, kalcij, titanij, krom, mangan, srebro	tamnosiva
Boron, klor	zelena
Jod	ljubičasta
Helij	svjetloroza

Pri bojanju aminokiselina najčešće korišteni sustavi boja su: *Clustal*, *Lesk*, *Cinema*, *Shapely*,... Boje aminokiselina korištene u ovom radu najsličnije su onima u Shapely sustavu. Svrha označavanja aminokiselina različitim bojama je mogućnost identifikacije aminokiselina u neobičnoj ili neočekivanoj okolini [3].

Korišteni sustav boja za označavanje lanaca u ovom radu definiran je prema

uzoru na postojeći alat za vizualizaciju molekula *Jmol*. Sustav boja definiran u Jmol-u posebno je koristan za razaznavanje dijelova multimerskih struktura ili individualnih DNK niti dvostruke uzvojnice [2]. Detaljniji popis boja korištenih u ovom radu može se naći u *Privitku A*.

2.3. Prikaz trodimenzionalnih objekata

OpenGL (skraćeno od eng. *Open Graphics Library*) je industrijski standard za grafiku visokih performansi [5]. OpenGL *API* (eng. *Application Programming Interface*) omogućuje korisniku jednostavnu izgradnju složenih sustava koji zahtijevaju grafičko ubrzanje, te time i komunikaciju s grafičkom karticom. Za prikazivanje grafike i iscrtavanje trodimenzionalnih objekata u ovome radu korišten je upravo OpenGL.

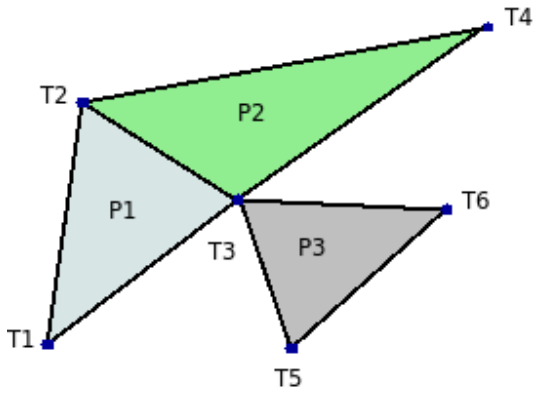
Kako bi se grafički prikaz ovog alata mogao iscrtavati i osvježavati relativno brzo, potrebno je definirati odgovarajuću strukturu podataka koja će sadržavati sve podatke o pojedinom trodimenzionalnom objektu. Jedna od takvih struktura sastoji se od dvije liste: liste točaka (eng. *vertices*) i liste poligona. Svaka točka definirana je svojim 3D koordinatama *(x, y, z)*, normalom *(n_x, n_y, n_z)*, bojom *(r, g, b, a)* i koordinatama teksture *(u, v)*. Poligoni su reprezentirani u obliku popisa indeksa točaka koje predstavljaju vrhove tog poligona: *(i_1, i_2, i_3, i_4,...)*. U ovome radu, poligoni su trokuti, što znači da se svaki od njih sastoji od točno tri cjelobrojna indeksa. Tablicama 2 i 3, te slikom 7 prikazana je upravo opisana struktura podataka korištena za konstrukciju trodimenzionalnih objekata.

Tablica 2. Lista svih točaka korištenih pri definiranju 3D objekta.

Lista točaka				
T_1	x_1, y_1, z_1	n_{x1}, n_{y1}, n_{z1}	r_1, g_1, b_1, a_1	u_1, v_1
T_2	x_2, y_2, z_2	n_{x2}, n_{y2}, n_{z2}	r_2, g_2, b_2, a_2	u_2, v_2
T_3	x_3, y_3, z_3	n_{x3}, n_{y3}, n_{z3}	r_3, g_3, b_3, a_3	u_3, v_3
T_4	x_4, y_4, z_4	n_{x4}, n_{y4}, n_{z4}	r_4, g_4, b_4, a_4	u_4, v_4
T_5	x_5, y_5, z_5	n_{x5}, n_{y5}, n_{z5}	r_5, g_5, b_5, a_5	u_5, v_5
T_6	x_6, y_6, z_6	n_{x6}, n_{y6}, n_{z6}	r_6, g_6, b_6, a_6	u_6, v_6

Tablica 3. Lista poligona definiranih pomoću indeksa točaka, navedenih u tablici 2.

Lista poligona	
P_1	T_1, T_2, T_3
P_2	T_2, T_3, T_4
P_3	T_3, T_5, T_6



Slika 7. Primjer slike nacrtane pomoću liste točaka i poligona.

2.4. Prianjanje proteina

Sustav Protein Docking Tool izvršava postupak prianjanja proteina razvojem njihove površine u red kuglinih funkcija. Ovu metodu razradio je i prvi implementirao Dave Ritchie 1998. godine [8]. PDT se sastoji od dvije važnije cjeline:

- *predocking* – obrađuje ulazne podatke i računa koeficijente površine
- *docking* – pretražuje prostor stanja te određuje konformaciju koja daje najbolji rezultat

2.4.1. Predocking

Prvi korak koji ovaj sustav izvršava je učitavanje i *parsiranje PDB* datoteka [9]. PDB je skraćenica od eng. *Protein Data Bank*, a predstavlja svjetski repozitorij za procesiranje i distribuciju podataka o trodimenzionalnim strukturama velikih molekula proteina i nukleinskih kiselina. Ovi su podatci dobiveni eksperimentalno, najčešće rendgenskom kristalografijom ili *NMR* spektroskopijom (NMR - nuklearna magnetska rezonanca). PDB je također i sinonim za format zapisa datoteka, koji sadrži sve potrebne informacije o pojedinoj makromolekularnoj strukturi (položaj atoma, podatke o lancima i aminokiselinama,...). Kako podatci o radijusima atoma nisu prisutni u zapisu PDB datoteka, potrebno ih je posebno odrediti.

Jednom kada su određeni položaji svih atoma kao i njihovi radijusi, moguće je odrediti površinu molekule na način opisan u poglavlju 2.1. Za samo određivanje površine PDT koristi program *MSMS.*

U trećem koraku, površina molekule se diskretizira pomoću grida, koji predstavlja trodimenzionalnu kocku dimenzija *N*x*N*x*N*, gdje je *N* dimenzija stranice kocke u angstremima. Svaka dimenzija kocke podijeljena je na manje dijelove određenih dimenzija koji se nazivaju *ćelije*. Pojedina ćelija predstavlja dio (diskretiziranog) prostora jednakih svojstava, a njezine dimenzije određuju finoću diskretizacije. U konkretnom slučaju kod PDT-a, vrijednost ćelije je realni broj koji predstavlja prisutnost dijela površine molekule u toj ćeliji. Ovakva reprezentacija površine potrebna je zbog numeričke integracije u izrazima za izračun koeficijenata kuglinih funkcija.

Kugline funkcije (eng. *spherical harmonics*, sferni harmonici) su zapravo dvodimenzionalne površine. Linearnom kombinacijom kuglinih funkcija moguće je prikazati svaku funkciju na kugli:

$$\mu(\Phi,\Theta)=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}a_{lm}y_{lm}(\Phi,\Theta)\,, \qquad (1)$$

gdje su $y_{lm}(\Phi,\Theta)$ kugline funkcije zadane sfernim koordinatama (Φ,Θ), a a_{lm} koeficijenti parametrizacije površine proteina. No, u PDT-u se ne primjenjuje samo

prikaz površine u obliku kuglinih funkcija upravo zbog njihove dvodimenzionalnosti. One se dodatno množe s odabranim radijalnim funkcijama, čime se ljuskama proteina definira volumen. Pogodne radijalne funkcije definirane su izrazom:

$$R_{nl}(r) = N_{nl} e^{-\rho/2} \rho^l L_{n+l}^{2l+1}(\rho), \qquad (2)$$

gdje su $L_{n+l}^{2l+1}(\rho)$ pridruženi Laguerrovi polinomi, ρ skalirana udaljenost središta, a N_{nl} normalizacijski koeficijent. Površinu proteina $\sigma(\underline{r})$ sada možemo prikazati pomoću razvoja u novoj bazi:

$$\sigma(\underline{r}) = \sum_{nlm} a_{nlm} R_n(\underline{r}) y_{lm}(\Phi, \Theta) = \sum_{nlm} a_{nlm} F_{nlm}(\underline{r}). \qquad (3)$$

Za svaki se protein određuju ukupno dva skupa koeficijenate: koeficijenti *unutarnje* i koeficijenti *vanjske ljuske*. Unutarnja ljuska definirana je kao volumen između molekularne površine i površine određene unutarnjom ovojnicom površinskih atoma, dok je vanjska ljuska volumen između molekularne površine i površine dostupne otapalu. Model s dvije ljuske potreban je zbog načina implementacije funkcije ocjenjivanja konformacija.

Upravo opisani postupak odvija se dva puta: jednom za protein ligand, te drugi put za protein receptor.

2.4.2. Docking

Ulaz u sam proces prianjanja proteina su 4 skupa koeficijenata – unutarnje i vanjske ljuske liganda i receptora. Pretraživanje prostora stanja izvodi se rotacijom proteina receptora za kuteve $(0, \beta_1, \gamma_1)$, te liganda za kuteve $(\alpha_2, \beta_2, \gamma_2)$. Ligand se i dodatno pomiče za radijus *R* po *Z*-koordinatnoj osi.

Funkcija korištena za ocjenjivanje kvalitete konformacije dana je slijedećom formulom:

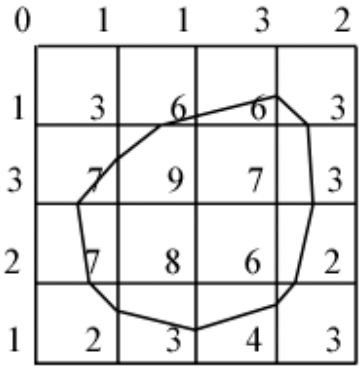
$$S = \int \rho_A(\underline{r}_A)\tau_B(\underline{r}_B)dV + \int \tau_A(\underline{r}_A)\rho_B(\underline{r}_B)dV + Q\int \tau_A(\underline{r}_A)\tau_B(\underline{r}_B)dV, \qquad (4)$$

gdje je ρ vanjska ljuska, τ unutarnja ljuska određenog proteina, a A i B su oznake proteina liganda i receptora. Faktor *Q* označava kaznu za prodiranje jednog proteina

u drugi. Prodiranje se kažnjava negativnim bodovima, jer je to fizikalno nemoguć slučaj. Izraz (4) koristiti će se u nastavku, prilikom ručnog ocjenjivana konformacija proteina i liganda.

2.5. Rekonstrukcija površine modelirane kuglinim funkcijama

Površinu proteina predstavljenu koeficijentima kuglinih funkcija moguće je rekonstruirati i prikazati. Kako bi se to ostvarilo, potrebno je stvoriti grid (sličan onome opisanom u poglavlju 2.4), čiji elementi će biti realne vrijednosti dobivene pomoću koeficijenata kuglinih funkcija. Elementi tako dobivenog grida predstavljaju *isovrijednosti*, koje određuju površinu koja prolazi tom točkom prostora. Na ovaj način ne može se eksplicitno prikazati površina proteina, već je potrebno zadati željenu *referentnu isovrijednost*, u odnosu na koju će se površina rekonstruirati. Interpolacijom između postojećih vrijednosti u gridu, rekonstruira se odabrana površina. Slika 8 prikazuje postupak rekonstrukcije na dvodimenzionalnom primjeru.



Slika 8. Dvodimenzionalni primjer određivanja površine uz određenu referentnu isovrijednost jednaku 5.

Svaka ćelija grida ima ukupno osam susjednih elemenata, čime za svaki njezin vrh dobivamo koordinate i određeni intenzitet (isovrijednost). Ta ćelija je zapravo najsitniji volumni element nekog modela, te se stoga naziva *voksel* (eng. *voxel*,

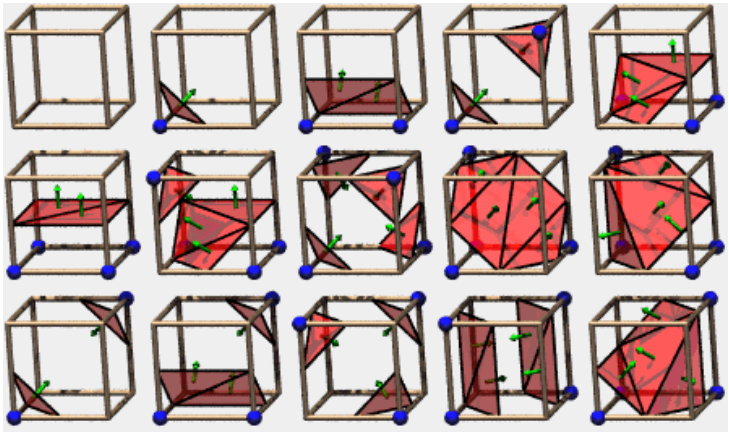
volume element).

Marching Cubes algoritam namijenjen je upravo određivanju isopovršine pomoću dane referentne vrijednosti na trodimenzionalnom gridu. On se zasniva na ideji interpolacije površine između vrijednosti koje padaju unutar i izvan volumena površine [1]. Za vrijedosti koje su manje od referentne, promatrana se točka nalazi izvan volumena opisanog površinom, dok se za veće vrijednosti točka nalazi unutar tog volumena. Pri tome se može koristiti bilo koja vrsta interpolacije. U ovom radu korištena je linearna interpolacija. Marching Cubes u pojedinom trenutku razmatra samo jedan voksel i vrijednosti intenziteta u njegovim točkama. Zatim je potrebno iterirati kroz sve volumne elemente, te tako odrediti ukupnu isopovršinu.

Kako se jedan voksel sastoji od 8 vrhova, to nam daje ukupno 256 potencijalnih kombinacija stanja u tim vrhovima. Algoritam se može pojednostaviti ako uzmemo u obzir kombinacije koje se ponavljaju pod određenim uvjetima:

- rotacija za proizvoljan broj stupnjeva oko bilo koje koordinatne osi,
- zrcaljenje oblika po bilo kojoj osi,
- invertiranjem stanja svih vrhova i okretanjem normala određenih poligona.

Zahvaljujući ovim uvjetima, početnih 256 kombinacija može se reducirati na samo 15 različitih. Slika 9 prikazuje tih 15 kombinacija, zajedno s predefiniranim skupovima poligona za te slučaje, namijenjenim aproksimaciji površine.



Slika 9. Ukupno 15 mogućih kombinacija stanja u vrhovima i njihova poligonizacija.

Na slici 9 plave kugle označavaju vrhove za koje je određeno da se nalaze unutar isopovršine, a zelene strelice označavaju normale odgovarajućih poligona. Određivanjem poligona svih voksela, te njihovim sakupljanjem u listu, dobivamo trodimenzionalni model isopovršine za odabranu referentnu isovrijednost.

2.6. Ocjenjivanje rezultata procesa prianjanja

Rezultati procesa prianjanja proteina prezentirani su u obliku rang liste konformacija, redom od najbolje do najlošije, po kriteriju opisanom u poglavlju 2.4. No, konformacija koja daje najbolje rezultate uspoređivanjem samo geometrijske komplementarnosti površina ne mora nužno odgovarati i kristalografskoj strukturi. Razlog tome je složenost stvarnog procesa koji se ovim putem modelira, a obuhvaća, osim geometrijske komplementarnosti, i elektrostatiku, hidrofobnost, te druga svojstva molekula. Zbog toga je u sklopu ovog rada razvijena metoda ocjenjivanja točnosti rezultata prianjanja proteina.

Za određivanje ocjene rezultata, potrebno je napraviti usporedbu dobivene konformacije sa stvarnom kristalografskom strukturom kompleksa. Ocjena točnosti određuje se pomoću udaljenosti središta atoma dobivene i kristalografske strukture, prema izrazu:

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N} \left| \vec{p}_{Kf(i)} - \vec{p}_{Di} \right|}{N}}, \qquad (5)$$

gdje je $\vec{p}_{Kf(i)}$ položaj *i*-tog atoma unutar kristalografske (točne) strukture molekule, $\vec{p}_{Di}$ položaj istog atoma unutar ispitivane konformacije, a *N* broj atoma konformacije. $f(i)$ predstavlja funkciju kojoj je ulaz indeks atoma u ispitivanoj konformaciji, a kao rezultat daje indeks istog atoma unutar strukture točnog kompleksa. Potreba za ovakvom funkcijom proizlazi iz toga da se PDB datoteke liganda i receptora razlikuju od PDB datoteke kompleksa. Stoga je prvo potrebno odrediti funkciju $f(i)$. Algoritam postupka ocjenjivanja točnosti rezultata tada izgleda ovako:

1. učitati molekule proteina (receptor, ligand i točan kompleks),
2. indeksirati atome liganda i receptora prema rednim brojevima istih atoma točnog kompleksa – ovo je nužan korak za naknadno postavljanje orijentacije,
3. učitati rang listu rezultata,
4. transformirati ligand i receptor da odgovaraju konformaciji *k*-tog rezultata,
5. orijentirati dobiveni kompleks u istom smjeru kao i kristalografsku strukturu,
6. izračunati RMSD,
7. ukoliko ima još rezultata za ocjenjivanje, skočiti na korak 3, u protivnome je ocjenjivanje završeno.

Za razumijevanje prethodnog algoritma potrebno je još razjasniti pojam orijentacije. Kako su receptor i ligand zarotirani za neke proizvoljne kutove i međusobno razmaknuti u prostoru, dobiveni kompleks gotovo se nikada neće poklapati (po apsolutnom kriteriju) s točnom kristalografskom strukturom. Apsolutni kriterij ovdje znači da strukture dobivenog i točnog kompleksa mogu biti identične, ali drugačije postavljene u prostoru. Stoga im je potrebno prilagoditi orijentacije.

Prilagodba orijentacija provodi se na temelju relativnog položaja tri atoma receptora, odnosno kompleksa. Kako smo već indeksirali iste atome receptora u odnosu na kompleks, vrlo jednostavno možemo izdvojiti koordinate triju atoma receptora koje moramo poklopiti s tri koordinate kompleksa. To se izvodi na slijedeći način:

1. odredi se translacijska matrica potrebna da se referentne točke (tri razmatrane točke) pomaknu za negativni položaj prve točke, tako da prva točka dođe u ishodište koordinatnog sustava,
2. odredi se rotacijska matrica potrebna da se vektor između druge i prve točke poklopi s pozitivnim smjerom *X*-osi,
3. odredi se rotacijska matrica oko *X*-osi, tako da se treća točka spusti na *XZ*-ravninu. U ovom slučaju postoje dva rješenja, jer se treća točka može poklopiti s *XZ*-ravninom i ako je zarotiramo za dodatnih 180°. Stoga se uvijek uzima ono rješenje koje daje rezultantnu treću točku čija je vrijednost *Z*-koordinate veća od 0,

4. izračunate transformacijske matrice se pomnože, te se primijene na sve koordinate atoma trenutnog proteina.

Gornji postupak provodi se prvo na točnom kompleksu, te zatim na receptoru. Transformacijska matrica receptora primjenjuje se još i za prilagođavanje orijentacije liganda. Time se osigurava očuvanost relativnih odnosa receptora i liganda. Kao rezultat ovog postupka, svi atomi receptora poklopljeni su s odgovarajućim atomima točnog kompleksa, a na RMSD utječe jedino položaj atoma liganda u odnosu na kompleks.

3. Općeniti opis sustava

Alat razvijen u sklopu ovog rada prvenstveno je namijenjen vizualizaciji pojedinih međukoraka procesa prianjanja proteina, te je integriran u postojeći sustav Protein Docking Tool. Alat (nazvan *Longview*) nudi pregršt mogućnosti, te je vrlo fleksibilan, što znači da korisnik ima gotovo potpunu slobodu određivanja što će se i kako prikazati.

Prikaz se ostvaruje preko strukture koja se zove *graf scene*. Graf se sastoji od hijerarhijski povezanih čvorova, od kojih svaki predstavlja određenu funkciju, kao što su: rotacija, translacija, boja, iscrtavanje objekata ili čak izvršavanje niza naredbi. Dio sustava dizajniran je posebno za ostvarivanje svih funkcija direktno povezanih s OpenGL-om. To osigurava logičku odvojenost funkcionalnog dijela Longview-a koji je pisan u *ANSI* C++-u (eng. *American National Standards Institute*) te je neovisan o platformi na kojoj se izvršava, i dijela namijenjenog za iscrtavanje (renderiranje, eng. *rendering*). Uz OpenGL, vrlo je popularan i DirectX grafički API (koji je specifičan samo za *Windows* operativni sustav). Upravo zbog opisane strukture alata, u nastavku njegovog razvoja biti će moguće relativno jednostavno napraviti podršku za renderiranje korištenjem DirectX-a (ili nekog drugog grafičkog API-ja).

Longview trenutno podržava učitavanje i prikaz sljedećih formata zapisa trodimenzionalnih modela:

- dat
    - tekstualni format, razvijen za potrebe ovog rada
    - sadrži zapis liste točaka i liste poligona
- ms
    - tekstualni format, sastoji se od dvije datoteke:
        - <ime>.vert – sadrži listu točaka
        - <ime>.face – sadrži listu poligona
    - predstavlja površinu molekule dobivene programom MSMS
- grid
    - binarni format

    - sadrži podatke (realne vrijednosti) o trodimenzionalnom gridu, koji reprezentira površinu molekule (stvarnu diskretiziranu površinu ili podatke o isopovršinama dobivenim iz koeficijenata sfernih harmonika)
- PDB (Protein Data Bank)
    - tekstualni format
    - sadrži sve podatke o kristalografskoj strukturi pojedinog proteina: koordinate atoma, svojstva, podatke o vezama...

Pojedina PDB datoteka može sadržavati više *modela*, pri čemu neki model, na primjer, predstavlja jednu dobivenu (potencijalno točnu) konformaciju procesa prianjanja proteina. U tom slučaju Longview omogućuje i odabir prikaza željenog modela.

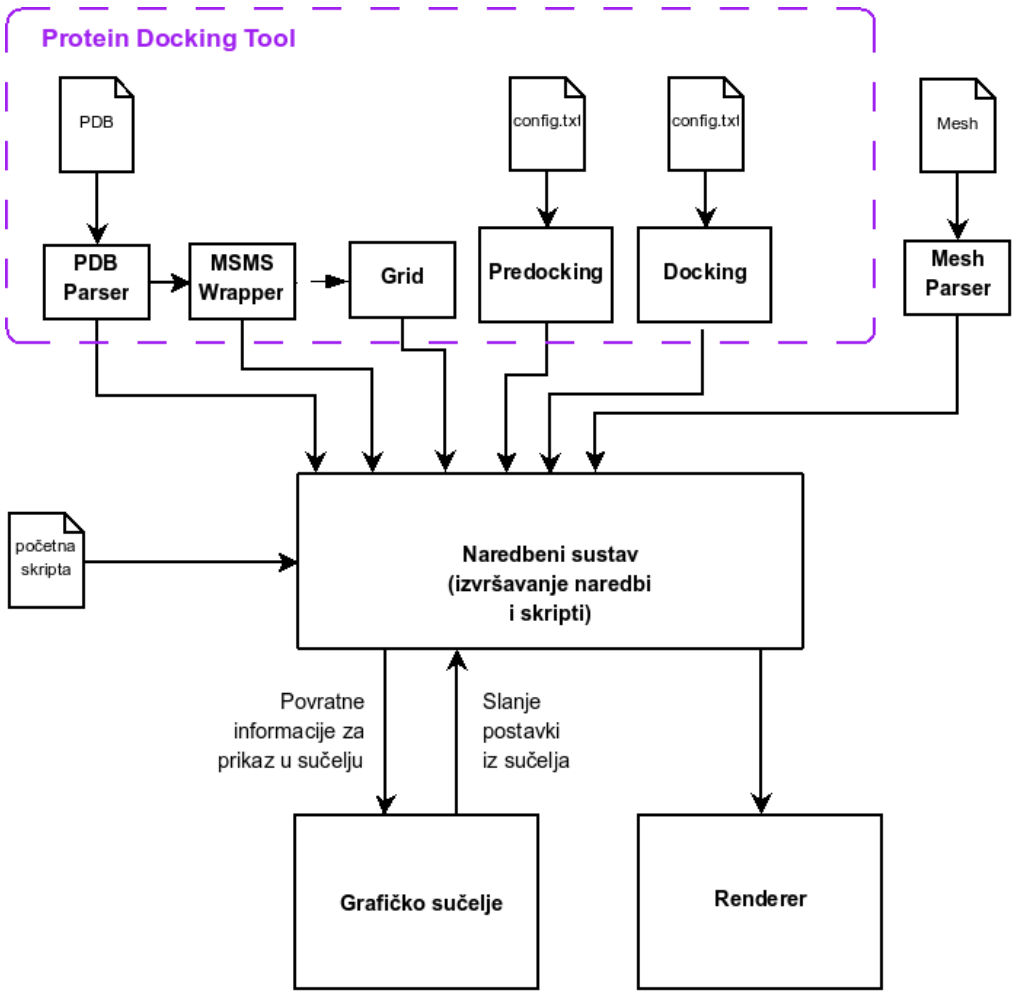
Svakom učitanom trodimenzionalnom objektu moguće je postavljati opcije prikaza, kao što su: boje (način bojanja, isključivanje bojanja), korištenje normala (sjenčanje objekta), odabir načina iscrtavanja (popunjeni ili žičani (eng. *wireframe*) prikaz), način prikaza molekula (kalotni, žičani, štapići, štapići i kugle), uključivanje/isključivanje iscrtavanja pojedinog objekta, i drugi. Sve učitane objekte, te načine nihova prikaza moguće je superponirati jedan na drugi, što pruža mogućnost usporedbe pojedinih objekata.

Upravljanje Longview-om ostvaruje se preko tekstualnih naredbi ili korisničkog sučelja. Naredbe je moguće izvršavati iz naredbenog retka u samom sučelju, ili preko skripti. Grafičko sučelje alata također je bazirano na skriptnim naredbama, tj. prilikom izvršavanja svih akcija potrebni podatci šalju se putem skriptnih naredbi u sustav za njihovu obradu i izvršavanje. Ovaj način dizajna kontrole nad sustavom nudi mogućnost vrlo moćnih dodatnih funkcija, kao što su spremanje radne okoline ili definiranje proizvoljne početne radne okoline.

Od dodatnih naprednijih mogućnosti i za razliku od ostalih alata slične namjene, Longview omogućuje i uključivanje do četiri odvojena prikaza, od kojih je svaki moguće zasebno konfigurirati kako bi prikazivao proizvoljnu scenu.

S programske strane, alat je realiziran na objektno orijentiran način u C++ programskom jeziku. Radi lakšeg i kvalitetnijeg povezivanja s PDT-om, Longview

koristi strukture podataka koje su otprije dostupne i korištene u PDT-u (npr. *PdbFile*, *Grid*,...), uz što definira i nekoliko novih (npr. *Mesh*, *Vector3*,...). Slika 10 predstavlja grafički blok prikaz alata Longview.



Slika 10. Grafički blok prikaz alata Longview.

4. Struktura alata

U ovom poglavlju objašnjeni su neki od osnovnih elemenata strukture alata Longview. Osim samog dizajna i organizacije koda, izloženi su i neki vrlo bitni dijelovi alata:

- graf scene – osnova za postavljanje prikaza
- skripte i naredbe – osnova za upravljanje alatom
- sučelje – osnova za jednostavnost korištenja

4.1. Organizacija kôda

Kôd alata Longview pisan je u programskom jeziku *C++*. Alat je dizajniran na objektno orijentiran način, što znači da su pojedini dijelovi sustava razlomljeni u manje cjeline – *klase*. Svaka klasa sadrži odgovarajuće članske (eng. *member*) varijable i funkcije, potrebne za ostvarivanje željenih funkcionalnosti i svojstava tih klasa. Kako bi se onemogućila nekontrolirana izmjena vrijednosti pojedinih varijabli, ili zabranilo izvršavanje pomoćnih funkcija, takve varijable i funkcije deklarirane su kao privatne. Osim ovakvog osnovnog oblika klasa, u Longview-u se koriste i *singleton* klase, kao i nasljeđivanje klasa. Singleton je klasa čiji se objekt može instancirati samo jednom. Prilikom svakog sljedećeg pristupa toj klasi, zapravo se pristupa objektu (preko pokazivača) koji je prvotno stvoren. Primjer singleton klasa koje su definirane u Longview alatu: `VmolCommand`, `VmolGui`, `VmolScene` i `Log`. Nasljeđivanje je način oblikovanja novih klasa korištenjem već prethodno definiranih klasa, a svrha mu je ponovno iskorištenje postojećeg koda uz čim manje modifikacija. Pri tome nova klasa preuzima postojeće varijable i funkcije definirane u klasi koju nasljeđuje. Ovakav način dizajna klasa koristi se u Longview-u prilikom oblikovanja pojedinih dijelova korisničkog sučelja (na primjer, klasa `TextInput` koja nasljeđuje klasu `Fl_Input`).

4.1.1. Osnovne klase

U tablici 4 naveden je opis važnijih klasa sadržanih u alatu Longview, dok tablica 5 opisuje pomoćne klase ovog alata. Slika 11 prikazuje dijagram klasa, gdje se mogu vidjeti njihove strukture i međusobni odnosi.

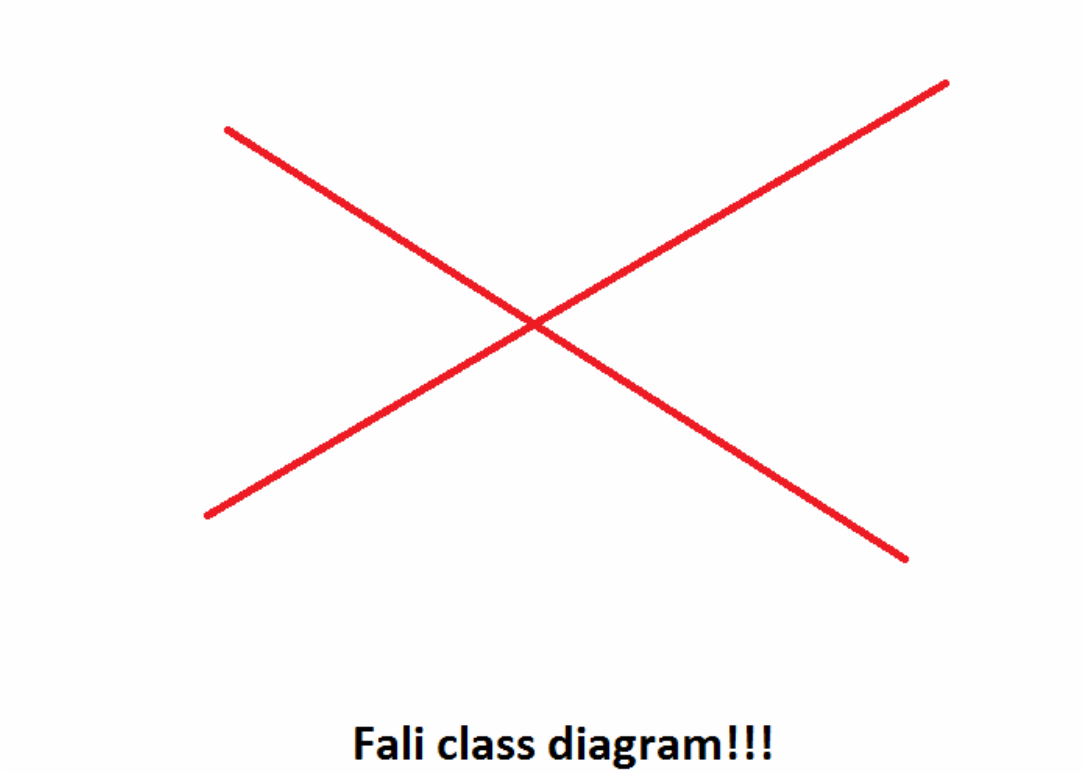
Tablica 4. Opis važnijih klasa u Longview-u.

Ime klase	Opis
`VmolCommand`	Obrađuje i izvršava skriptne naredbe, te čuva trenutno stanje radne okoline.
`VmolGui`	Služi za stvaranje i osvježavanje grafičkog sučelja, isto kao i za izvršavanje korisničkih radnji povezanih sa sučeljem.
`VmolSceneNode`	Objekt ove klase predstavlja jedan čvor grafa scene.
`VmolScene`	Služi za iscrtavanje trodimenzionalnih objekata, te izvršavanja radnje pojedinog čvora, isto kao i cijelog grafa scene.
`VmolView`	Klasa koja opisuje dio sučelja u kojem se iscrtava prikaz.
`Log`	Ostvaruje bilježenje poruka u zapisnik, te obrađivanje poruka o pogreškama.
`Mesh`	Klasa za učitavanje i skladištenje podataka o pojedinom trodimenzionalnom modelu (grid, molekularna površina, obični model).

Tablica 5. Opis pomoćnih klasa u Longview-u.

Ime klase	Opis
`TextInput`	Klasa bazirana na Fl_Input klasi. Predstavlja kontrolu za unost teksta, proširena mogućnostima za kopiranje i ispuštanje teksta (eng. *copy/paste*).
`CommandTextInput`	Isto kao i `TextInput`, uz dodatno svojstvo da se pritiskom na tipku *enter* upisani tekst interpretira kao naredba skriptnog jezika.

`Parameters`	Pomoćna klasa koja sadrži vrijednosti svih parametara prepoznatih iz naredbi skriptnog jezika.
`VariableName`	*Template* klasa koja povezuje proizvoljni tip podatka s imenom (objekt tipa `string`). Koristi se za skladištenje objekata stvorenih naredbama skriptnog jezika.
`ControlPosition`	Sadrži trenutni položaj (`x`, `y`), te visinu (`width`) i širinu (`height`) za pojedinu kontrolu. Svi podatci su tipa `int`.
`MouseEventBinds`	Struktura sadrži pokazivače na varijable koji su povezani s određenom radnjom miša. Sastoji se od dva spremnika tipa `vector`, od kojih jedan sadrži pokazivač na odabranu varijablu, a drugi vrstu varijable na koju prvi pokazuje. Radnja miša i odabrana varijabla povezani su preko indeksa varijable u spremniku.
`MsVertexInfo`	Sadrži dodatne informacije o svakoj točki površine, učitane iz *.vert* datoteke, stvorene od strane programa MSMS.
`MsFaceInfo`	Sadrži dodatne informacije o svakom poligonu površine, učitanom iz *.face* datoteke, stvorene od strane programa MSMS.
`Vector3`	Sadrži tri parametra (`x`, `y`, `z`) tipa `double`, te definirane razne operatore. Primjenjuje se, primjerice, za čuvanje koordinata određene točke modela.
`EulerVector3`	Sadrži tri parametra (`alpha`, `beta`, `gamma`) tipa `double`. Primjenjuje se priliko određivanja rotacije po Eulerovim kutovima.
`Color`	Sadrži tri parametra (`r`, `g`, `b`, `a`) tipa `unsigned char`. Primjenjuje se za čuvanje podataka o pojedinoj boji.
`Face3`	Sadrži tri parametra (`index[0]`, `index[1]`, `index[2]`) tipa `unsigned int`. Primjenjuje se za čuvanje indeksa točaka koje čine određeni poligon.



Slika 11. Dijagram klasa alata Longview.

4.1.2. Podjela u manje sustave i sučelja između njih

Alat Longview može se podijeliti na nekoliko važnijih manjih sustava: naredbeni sustav, sustav prikaza, sustav grafičkog sučelja, te sustav za vođenje zapisnika i obrađivanje pogrešaka.

Naredbeni sustav

Ulazi:

- naredba u obliku tekstualnog retka
- skripta s naredbama, predana u obliku putanje do datoteke koja sadrži skriptu

Izlazi:

- ispis izvršenih naredbi u skriptnu datoteku
- rezultati izvođenja naredbi pohranjeni u zapisnik

Opis:

Sustav je namijenjen izvršavanju naredbi zadanih u obliku tekstualnog retka. Osim toga, moguće je izvršavati i skripte, koje predstavaljaju niz tekstualnih naredbi za uzastopno izvršavanje. Naredbeni sustav vodi brigu o objektima koji se nalaze u radnoj okolini, te se povezuje sa svim ostalim sustavima radi ostvarivanja funkcionalnosti alata preko postojećih resursa.

Sustav prikaza

Ulazi:

- trodimenzionalni objekti za prikaz (mesh, PDB)
- izvorni čvor grafa scene

Izlazi:

- prikaz rezultata iscrtavanja u prozoru alata
- dodavanje podataka o pogreškama u zapisnik

Opis:

Sustav je namijenjen ostvarivanju trodimenzionalnog prikaza modela i molekula. On omogućuje iscrtavanje pojedinog modela, ili cijelog grafa scene. Graf scene predaje se sustavu preko svojeg izvorišnog čvora. Izvorišni čvor se iscrtava, te se zatim rekurzivno prelazi i na iscrtavanje njegove djece.

Sustav grafičkog sučelja

Ulazi:

- postavke predane od naredbenog sustava
- stanje kontrola na sučelju (pritisnuti gumbi, promijenjen tekst,...)

Izlazi:

- prikaz sučelja alata na ekranu
- naredbe za naredbeni sustav

Opis:

Sustav služi za postavljanje izgleda i funkcionalnosti grafičkog sučelja. Pojedine kontrole sučelja postavljene su fiksno, ali za popunjavanje dinamičkih izbornika potrebna je komunikacija s naredbenim sustavom. Promjenom stanja kontrola na sučelju, sustav grafičkog sučelja pripremi skriptnu naredbu koja će ostvariti određenu funkcionalnost, te je na izlazu proslijedi naredbenom sustavu.

Sustav zapisnika i pogrešaka

Ulazi:

- putanja zapisnika
- tekst koji treba dodati u zapisnik
- podatci o pogrešci koja se dogodila (i treba biti dodana u zapisnik te obrađena)

Izlazi:

- datoteka zapisnika

- standardni izlaz za pogreške (*stderr*, ispis poruke o pogrešci na ekran)

Opis:

Svrha ovog sustava je bilježenje dolaznih poruka u zapisnik (datoteku u memoriji). Na ulaz sustava može doći i poruka o pogrešci, koja pri tome mora biti obrađena. Sve poruke se pohranjuju u zapisnik, dok se poruke o pogrešci dodatno ispisuju i na standardni izlaz za pogreške.

4.1.3. Dizajn podataka

Ulazni podatci u cijeli sustav dani su u obliku datoteka: *PDB*, *dat*, *grid*, *vert* i *face*, te *vsc*. Uz svaku od ovih datoteka vezane su određene klase i funkcije zadužene za njihovo učitavanje i obradu. Ukoliko se, nakon učitavanja, određeni podatci trebaju sačuvati u memoriji, koristi se jedan od standardnih spremnika u C++-u: `vector`. Ovakav spremnik služi za dinamičko upravljanje memorijom, kao što su dodavanje ili micanje elemenata iz liste, proširivanje veličine spremnika na željenu veličinu i prije popunjavanja korisnim elementima,... U `vector` spremniku elementi su pohranjeni slijedno u memoriji. Uprvo zbog tih svojstava, `vector` je u Longview-u korišten za skladištenje podataka o trodimenzionalnim i drugim strukturama.

PDB datoteke

Općeniti opis PDB datoteka dan je u poglavlju 2.4.1. Tipična PDB datoteka koja opisuje protein sastoji se od nekoliko stotina, pa čak do tisuća linija teksta. Primjerak PDB datoteke može sadržavati velik broj definiranih ključnih riječi. No, usprkos tome, neke od njih se češće koriste od ostalih, a neke su nam opet od bitnijeg značaja za određivanje svojstava molekula. Dio tipične strukture PDB datoteke u kojoj su zapisani podatci o pojedinim atomima prikazan je slikom 12.

```
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
MODEL        1
ATOM      1  N   ALA A   1      11.104   6.134  -6.504  1.00  0.00           N
ATOM      2  CA  ALA A   1      11.639   6.071  -5.147  1.00  0.00           C
...
...
...
ATOM    293 1HG  GLU A  18     -14.861  -4.847   0.361  1.00  0.00           H
ATOM    294 2HG  GLU A  18     -13.518  -3.769   0.084  1.00  0.00           H
TER     295      GLU A  18
ENDMDL
MODEL        2
ATOM    296  N   ALA A   1      10.883   6.779  -6.464  1.00  0.00           N
ATOM    297  CA  ALA A   1      11.451   6.531  -5.142  1.00  0.00           C
...
...
ATOM    588 1HG  GLU A  18     -13.363  -4.163  -2.372  1.00  0.00           H
ATOM    589 2HG  GLU A  18     -12.634  -3.023  -3.475  1.00  0.00           H
TER     590      GLU A  18
ENDMDL
```

Slika 12 . Dio tipične PDB datoteke.

Prvi i drugi redak predstavljaju redni broj pojedinog znaka u zapisu, dok se u samoj strukturi PDB datoteka oni ne nalaze. Kako su nam za određivanje strukture molekule (npr. za vizualizaciju) najbitniji podatci o pojedinom atomu, na slici 13 prikazana je struktura *ATOM* linije, na način na koji je definira PDB standard.

COLUMNS	DATA TYPE	FIELD	DEFINITION
1 - 6	Record name	"ATOM"	
7 - 11	Integer	serial	Atom serial number.
13 - 16	Atom	name	Atom name.
17	Character	altLoc	Alternate location indicator.
18 - 20	Residue name	resName	Residue name.
22	Character	chainID	Chain identifier.
23 - 26	Integer	resSeq	Residue sequence number.
27	AChar	iCode	Code for insertion of residues.
31 - 38	Real(8.3)	x	Orthogonal coordinates for X in Angstroms.
39 - 46	Real(8.3)	y	Orthogonal coordinates for Y in Angstroms.
47 - 54	Real(8.3)	z	Orthogonal coordinates for Z in Angstroms.
55 - 60	Real(6.2)	occupancy	Occupancy.
61 - 66	Real(6.2)	tempFactor	Temperature factor.
77 - 78	LString(2)	element	Element symbol, right-justified.
79 - 80	LString(2)	charge	Charge on the atom.

Slika 13. Struktura ATOM linije PDB datoteke.

PDB datoteke za Longview učitava i obrađuje klasa `PdbFile`, koja je zapravo dio Protein Docking Tool sustava.

Grid datoteke

Grid se može zamisliti kao trodimenzionalna matrica popunjena realnim brojevima. Matrica je kubna, što znači da su njezina širina, visina i dubina jednake. Primjena ovakve strukture orijentirana je prema vizualizaciji površine molekule dobivene aproksimacijom pomoću razvoja u red sfernih harmonika. Format zapisa datoteke je binaran, a opisan je tablicom 6, u kojoj svaki redak predstavlja podatak određene veličine, slijedno zapisan u datoteci.

Tablica 6. Format zapisa .grid datoteke.

Veličina podatka (bajti)	Tip podatka	Opis
`4`	unsigned int	Broj elemenata grida (ćelija), označimo je s *N*.
`8`	double	Veličina stranice grida.
`8`	double	Veličina ćelije grida.
N `x 32`	double	Svaka ćelija sastoji se od 32 bajta podataka: (`x`, `y`, `z`) koordinate ćelije, i `v` vrijednosti te ćelije.

Grid datoteke učitava i obrađuje klasa `Grid`, koja je dio Protein Docking Tool sustava.

Dat, vert i face datoteke

Dat datoteke sadrže podatke o trodimenzionalnoj strukturi objekata. Alat Longview ih koristi, primjerice, za čitavanje modela kugli, štapića i koordinatnog sustava. Sastoje se od dvije glavne cjeline: liste točaka i liste poligona. Interpretacija

ovih listi dana je u poglavlju 2.3. Format zapisa datoteke je tekstualni, a njegova struktura prikazana je tablicom 7.

Tablica 7. Struktura .dat datoteke.

Podatak	Tip podatka	Opis
N	unsigned int	Broj točaka u modelu.
M	unsigned int	Broj poligona u modelu.
x_0 y_0 z_0 n_{x0} n_{y0} n_{z0} x_1 y_1 z_1 n_{x1} n_{y1} n_{z1} ... $x_{(N-1)}$ $y_{(N-1)}$ $z_{(N-1)}$ $n_{x(N-1)}$ $n_{y(N-1)}$ $n_{z(N-1)}$	double	Svaki redak predstavlja jednu točku zadanu svojim koordinatama (x, y, z) i normalom (n_x, n_y, n_z).
$i_{0,0}$ $i_{0,1}$ $i_{0,2}$ $i_{1,0}$ $i_{1,1}$ $i_{1,2}$ ... $i_{(M-1),0}$ $i_{(M-1),1}$ $i_{(M-1),2}$	unsigned int	Svaki redak predstavlja jedan poligon, zadan s tri točke, čiji su indeksi (i_0, i_1, i_2).

Vert i *face* datoteke predstavljaju izlazne datoteke programa MSMS, a oblika su analognog *dat* datotekama, uz razlamanje podataka u dvije cjeline. Liste točaka zapisane su u jednu datoteku (*vert*) , dok su poligoni zapisani u drugu (*face*). Format *vert* datoteke prikazan je tablicom 8, a format *face* datoteke tablicom 9.

Tablica 8. Format zapisa .vert datoteke.

Podatak	Tip podatka	Opis
N, S, D, R	unsigned int	N je broj točaka u modelu, S broj kugli, D gustoća, a R radijus sonde.
x_0 y_0 z_0 n_{x0} n_{y0} n_{z0} d_0 a_0 r_0 x_1 y_1 z_1 n_{x1} n_{y1} n_{z1} d_1 a_1 r_1 ... $x_{(N-1)}$ $y_{(N-1)}$ $z_{(N-1)}$ $n_{x(N-1)}$ $n_{y(N-1)}$ $n_{z(N-1)}$ $d_{(N-1)}$ $a_{(N-1)}$ $r_{(N-1)}$	double	Svaki redak predstavlja jednu točku zadanu svojim koordinatama (x, y, z) i normalom (n_x, n_y, n_z) kao i kod *dat* datoteka. Dodatni parametri su vrijednost d, indeks atoma a kojem točka površine pripada te radijus sonde r.

Tablica 9. Format zapisa .face datoteka.

Podatak	Tip podatka	Opis
M, S, D, R	unsigned int	M je broj poligona u modelu, S broj kugli, D gustoća, a R radijus sonde.
$i_{0,0}$ $i_{0,1}$ $i_{0,2}$ $p_{0,0}$ $p_{0,1}$ $i_{1,0}$ $i_{1,1}$ $i_{1,2}$ $p_{1,0}$ $p_{1,1}$ ... $i_{(M-1),0}$ $i_{(M-1),1}$ $i_{(M-1),2}$ $p_{(M-1),0}$ $p_{(M-1),1}$	3 x unsigned int, 2 x int	Svaki redak predstavlja jedan poligon, zadan s tri točke, čiji su indeksi (i_0, i_1, i_2) kao i kod *dat* datoteka. Svaki redak sadrži i dodatna dva parametra (p_0, p_1).

Dat, *vert* i *face* datoteke učitava i obrađuje klasa Mesh.

Vsc datoteke

Longview-ove skripte zapisuju se i učitavaju iz *vsc* datoteka. Datoteke su tekstualnog oblika, gdje svaki redak predstavlja jednu (ili više) naredbi skripnog jezika. Prilikom učitavanja datoteka, učitava se redak po redak, koji se istovremeno interpretira. Detaljniji opis skriptnih naredbi dan je u poglavlju 4.4. Vsc datoteke učitava i obrađuje klasa VmolCommand.

4.2. Povezanost s vanjskim aplikacijama

Longview omogućuje korištenje nekih funkcionalnosti koje nisu definirane u samom alatu. To se postiže povezivanjem Longview-a s vanjskim aplikacijama, i to:

- *MSMS* – kratica od *Michel Sanner Molecular Surface*
- *PDT* – kratica od *Protein Docking Tool*

Program MSMS koristi se prilikom određivanja molekularne površine proteina. Sam MSMS realiziran je kao izvršni program smješten u jednom od poddirektorija alata. Povezanost između Longview-a i MSMS-a ostvaruje se preko ulaznih i izlaznih datoteka, dok se program poziva pomoću `system` naredbe iz funkcija alata.

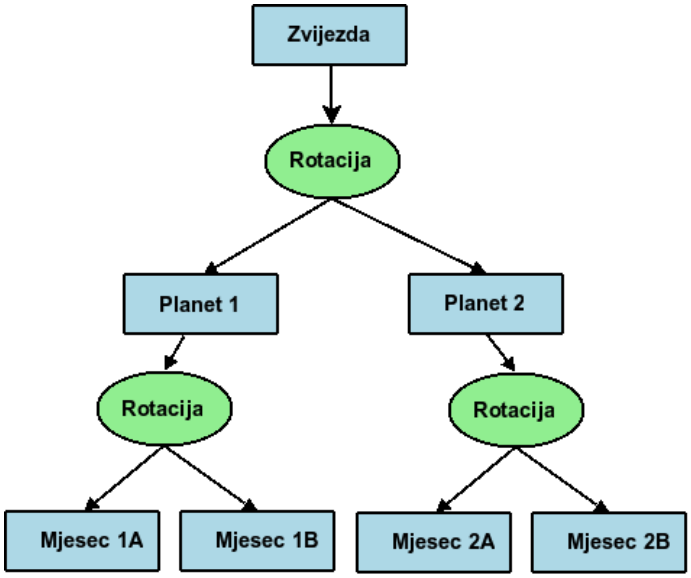
Protein Docking Tool je, s druge strane, realiziran u sklopu alata Longview. Nove datoteke s izvršnim kodom ovog alata dodane su u postojeći projekt PDT-a, te je kreirana nova postavka izgradnje izvršne datoteke (eng. *build configuration*). Svrha ove postavke je kompajliranje samo alata Longview, ali ne i izvršnih funkcija PDT-a (npr. *predocking* i *docking*). Izvršavanje pojedinih funkcija PDT-a ili stvaranje objekata iz klasa koje su tamo definirane, postiže se njihovim jednostavnim pozivanjem odnosno instanciranjem, što je omogućeno integracijom izvornih kodova sustava. Time je ostvareno sučelje između funkcionalnosti alata Longview i sustava PDT.

4.3. Graf scene

Graf scene je hijerarhijski način organizacije podataka. U svom najjednostavnijem obliku, ovaj graf predstavlja stablo u kojem roditeljski čvorovi utječu samo na ponašanje svoje djece. Dubina (tj. broj razina) takvog stabla je neograničen, kao i broj djece u svakom čvoru. No, za razliku od običnih stabala, kakva su primjerice definirana u teoriji grafova, grafovi scene su nešto složeniji. Svaki od čvorova predstavlja određenu akciju koja se mora izvršiti prije nego što se prijeđe na sljedeći čvor.

Radi lakšeg razumijevanja zašto su grafovi scene tako korisni, možemo razmotriti jednostavan primjer u kojem želimo napraviti model Sunčevog sustava, koji se sastoji od dva planeta, a svaki od planeta ima po dva mjeseca. Ovakav problem možemo

riješiti na dva načina: izgradnjom složene funkcije ponašanja za svako tijelo u našem Sunčevom sustavu ili izgradnjom grafa scene [6]. Odaberemo li ponašanje opisati pomoću fiksnih funkcija, a u međuvremenu primjerice odlučimo mijenjati položaje planeta, morat ćemo promijeniti potencijalno velik dio koda. Graf scene nam s druge strane omogućuje da problem riješimo malo apstraktnije, čime bismo omogućili veliku fleksibilnost ovog sustava. Slika 14 prikazuje mogući graf scene za opisani Sunčev sustav.



Slika 14. Model Sunčevog sustava predstavljen grafom scene.

U ovom slučaju, želimo li promijeniti ponašanje našeg sustava na način da *Planet 1* bude na drukčijoj udaljenosti od zvijezde nego u početnom slučaju, jednostavno dodamo čvor za translaciju prije iscrtavanja samog planeta. Ta radnja utjecati će i na mjesece *Planeta 1*, jer su sve transformacije primijenjene na pojedini čvor odmah prenesene i na njegovu djecu.

Sama funkcionalnost grafa scene nije fiksna, no funkcionalnost pojedinih čvorova jest. Ipak, osnovni građevni blokovi moraju biti definirani prije nego počnemo graditi scenu. U nastavku su navedeni svi čvorovi koje graf scene u Longview-u podržava, dok su programski oni izvedeni kao objekti klase `VmolSceneNode`. Uz taj popis, dan je i opis funkcionalnosti pojedinog čvora, te potrebni parametri.

4.3.1. NOP

Naziv je skraćenica od eng. *No Operation.* To je čvor koji ne izvršava nikakvu funkciju. Njegova je svrha samo da prenese funkcionalnost prethodnog čvora na svoju djecu.
Parametri: ovaj čvor nema parametara.

4.3.2. Mesh

Mesh čvor služi za iscrtavanje trodimenzionalnih modela, kao što su kugle, štapići i drugi. Na temelju *mode* parametra određuje se način i postavke iscrtavanja. Struktura mode parametra biti će objašnjena u nastavku rada.
Parametri:

- objekt (eng. *mesh*) koji se iscrtava, tip podatka: `Mesh`
- mode parametar, tip podatka: `unsigned long long int`

4.3.3. PDB

Služi za prikaz molekula učitanih iz PDB datoteka. Na temelju *mode* parametra određuje se način i postavke iscrtavanja.
Parametri:

- PDB objekt koji se iscrtava, tip podatka: `PdbFile`
- mesh objekt kugle za iscrtavanje atoma, tip podatka: `Mesh`
- mesh objekt štapića za iscrtavanje veza između atoma, tip podatka: `Mesh`
- mode parametar, tip podatka: `unsigned long long int`

4.3.4. Boja

Mijenja boju koja se trenutno koristi prilikom iscrtavanja.
Parametri:

- nova boja, tip podatka: `Color`

4.3.5. Pogled

Služi za mijenjanje trenutnih postavki pogleda (kamere).

Parametri:

- točka koja određuje položaj iz kojeg gledamo, tip podatka: `Vector3`
- točka u koju gledamo, tip podatka: `Vector3`
- vektor koji označava smjer "gore", tip podatka: `Vector3`

4.3.6. Pomak

Mijenja trenutnu transformacijsku matricu s novom operacijom translacije.

Parametri:

- vrijednost pomaka po sve tri koordinatne osi, tip podatka: `Vector3`

4.3.7. Rotacija oko X, Y i Z osi u Kartezijevom sustavu

Mijenja trenutnu transformacijsku matricu s novom operacijom rotacije po jednoj od tri koordinatne osi.

Parametri:

- vrijednost rotacije oko jedne osi, tip podatka: `double`

4.3.8. Rotacija po Eulerovim kutovima

Izvršava rotaciju po Eulerovim kutovima.

Parametri:

- vrijednost tri kuta rotacije (alfa, beta, gama), tip podatka: `EulerVector3`

4.3.9. Skaliranje

Skalira objekte koji slijede za određeni faktor po svakoj od koordinatnih osi.

Parametri:

- faktor skaliranja po koordinatnim osima, tip podatka: `Vector3`

4.3.10. Skripta

Izvršava sve naredbe iz skripte zadane svojom putanjom. Postupak izvršavanja skripti biti će detaljnije objašnjen u nastavku.

Parametri:

- putanja do skripte koja se treba izvršiti, tip podatka: `string`

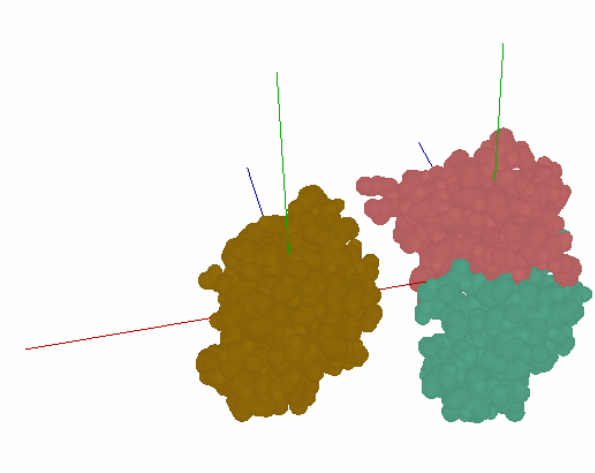
4.3.11. Modovi prikaza mesh i PDB modela

Mode nekog čvora je varijabla čiji bitovi predstavljaju postavke i svojstva prikaza pojedinog mesh-a ili molekule. Svaki mode je tipa unsigned long long int, što znači da sadrži ukupno 64 mogućih vrijednosti (bita). Tablica 10 prikazuje koje svojstvo je povezano s kojim bitom toga podatka. Ako je bit postavljen u stanje *1*, svojstvo je uključeno, dok je u protivnome isto svojstvo isključeno.

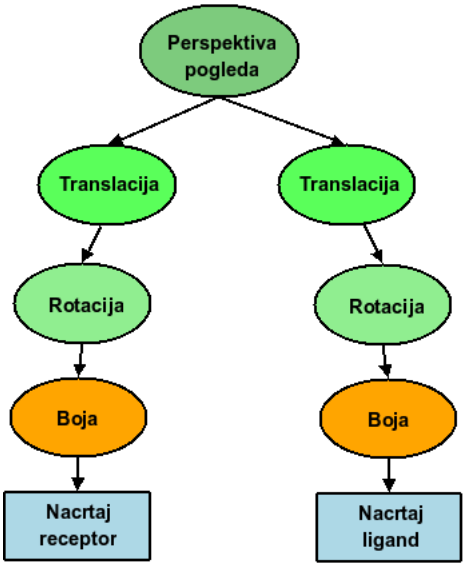
Tablica 10. Svojstva koja određuju bitovi pojedine mode varijable.

Bit	Svojstvo	Bit	Svojstvo
0	Prikaz geometrije	11	Žičani prikaz molekula
1	Primjena boja	12	Bojanje po atomima
2	Primjena normala	13	Bojanje po aminokiselinama
3	*Nedefinirano*	14	Bojanje po lancima
4	*Nedefinirano*	15	*Nedefinirano*
5	Primjena osvjetljenja	16	Iscrtavanje centra objekta
6	Žičani prikaz modela	17	Iscrtavanje granica modela
7	*Nedefinirano*	18	*Nedefinirano*
8	Kalotni prikaz	19	Pretpostavljena (*default*) svojstva
9	Prikaz štapićima	20 – 47	*Nedefinirano*
10	Prikaz štapićima i kuglama	48 – 63	Indeks modela iz PDB-a

Primjer prikaza dobivenog upotrebom grafa scene u Longview-u dan je na slici 15, dok slika 16 prikazuje istu scenu simbolički u obliku stabla. Učitani proteini su prvo centrirani prema svojoj masi u središte koordinatnog sustava, te potom razmaknuti po X-osi za 20Å, odnosno -20Å. Na prikazu proteina superponirana su i središta masa molekula, kako bi se mogla bolje predočiti njihova udaljenost.



Slika 15 . Prikaz dva proteina u Longview-u, razmaknutih po X-osi. Na slici su ucrtana i njihova središta.



Slika 16. Simbolički blok prikaz grafa scene sa slike 15.

4.4. Skripte i naredbe

Prema uzoru na postojeće alate slične namjene, kao što su *Pymol* i *Jmol*, Longview također omogućuje korisniku zadavanje naredbi preko naredbenog retka. Kako je Pymol izrađen u Python-u, izvedba interpretacije tekstualnih naredbi je relativno jednostavna zbog same funkcionalnosti tog skriptnog jezika. Longview je, s druge strane, rađen u *C++*-u, što znači da je za omogućavanje izvršavanja naredbi bilo potrebno dizajnirati cijeli novi skriptni jezik.

Prvi korak obrade naredbi je njihovo razdjeljivanje na sastavne dijelove. Nakon toga, raspoznaju se određene naredbe, podnaredbe i parametri o kojima ovisi ponašanje naredbenog sustava. Sam naredbeni sustav izveden je kao *singleton* klasa u C++-u (nazvana `VmolCommand`), što znači da može postojati samo jedna instanca objekta te klase. Time se omogućuje ostalim dijelovima sustava pristup zajedničkim resursima (tj. stvorenim objektima, varijablama, svojstvima ili korisnim funkcijama). Zbog svoje uloge, može se reći kako je naredbeni sustav zapravo *središnji dio* ovog alata za vizualizaciju, te time i jedna od njegovih *najbitnijih mogućnosti*. Neke od najvažnijih funkcija `VmolCommand` klase su:

- funkcija za izvršavanje pojedine naredbe, zadane preko parametra `commandString` u obliku stringa

```
int executeCommand(std::string commandString);
```

- funkcija za izvršavanje skripte s putanje zadane preko parametra `path`.

```
int loadScript(std::string scriptPath);
```

- funkcija za pronalaženje varijable (mesh-a, pdb-a, pogleda ili čvora) određenog tipa (parametar `variableType`) i zadanog imena (`variableName`). Povratna vrijednost je pokazivač na traženu varijablu, ili vrijednost `NULL` u slučaju da ona ne postoji.

```
void* findVariable(unsigned long int variableType, std::string
                                    variableName);
```

- funkcija za pronalaženje varijable koja osim pokazivača na tu varijablu vraća i njezin tip.

```
void* findVariable(std::string variableName, unsigned long int
                                    *retVariableType);
```

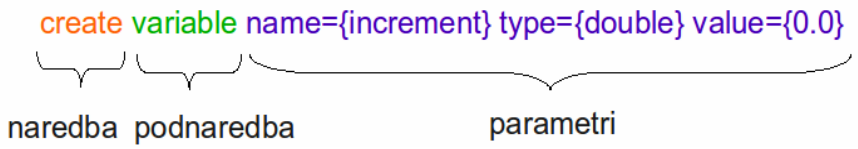
- funkcija za pronalaženje imena i tipa varijable koja je određena pokazivačem.

```
std::string findVariable(void *variablePointer, unsigned long
                         int *retVariableType);
```

Struktura naredbi, te moguće naredbe, podnaredbe i parametri dani su u nastavku.

4.4.1. Struktura naredbe

Primjer izgleda jedne naredbe prikazan je slikom 17.



Slika 17. Struktura naredbe u Longview-u.

Svaka naredba sastoji se od tri osnovna dijela:

1. naredbe
2. podnaredbe
3. parametara

Pri tome se svaki parametar može razdijeliti na dvije cjeline:

1. ime parametra
   - prije znaka '='
2. vrijednost parametra
   - nakon znaka '=', može sadržavati i više riječi ili vrijednosti, u kojem slučaju je potrebno koristiti vitičaste zagrade
   - primjer: želimo li kreirati varijablu tipa `string` (tekstualni niz znakova), to je moguće napraviti na sljedeći način
     - `create variable name={dobraProba1} type={string} value=Rijec`

        - `create variable name={dobraProba2} type={string} value={Ovo je recenica}`
        - `create variable name={losaProba3} type={string} value=Ovo je recenica`
    - u trećem primjeru pojedine će riječi, za koje očekujemo da budu sastavni dio varijable, zapravo biti prepoznate kao parametri, ali bez nekog posebnog značenja, te time i ignorirani, a vrijednost varijable `losaProba3` biti će jednaka "Ovo"

Naredba, podnaredba i svi parametri međusobno se odvajaju znakom razmaka. Treba napomenuti kako je u istoj naredbenoj liniji moguće izvršiti i neograničen broj naredbi, tako da se svaka pojedina naredba odvoji znakom '`;`'. Primjer:

```
create variable name={dobraProba1} type={string} value=Rijec;
create variable name={dobraProba2} type={string} value={Ovo je
recenica}
```

Uz sve navedeno, Longview omogućuje i korištenje *predložaka* (eng. *template*) u naredbama. Predlošci su po svojoj definiciji najsličniji pretprocesorskim naredbama u programskim jezicima kao C++. Oni služe za "umetanje" vrijednosti pojedinih varijabli na točno mjesto gdje su pozvani. Slični su pretprocesorskim naredbama upravo zbog toga što se prije izvršavanja svake naredbe prvo napravi zamjena predložaka s njihovom vrijednošću, a tek nakon toga ta naredba se izvrši. Kao predložak može se koristiti bilo koja postojeća varijabla tako da se ispred njezinog imena stavi znak '`:`'. Slijedi primjer korištenja predložaka:

```
create variable name={broj} type={long} value={123};
create variable name={tekst} type={string} value={:broj}
```

Nakon izvršavanja navedenih naredbi, vrijednost varijable `tekst` bit će jednaka "`123`".

Tablice 11 do 13 prikazuju sve naredbe, podnaredbe i parametre koje je moguće koristiti pri zadavanju naredbi u Longview alatu za vizualizaciju.

Tablica 11 . Opis mogućih naredbi u Longview-u.

Naredba	Opis
`create`	Stvaranje novih varijabli i objekata (float, double, pdb, node,...).
`change`	Mijenjanje parametara pojedinih varijabli i objekata.
`math`	Interpretacija i evaluacija matematičkih izraza.
`remove`	Brisanje varijabli i objekata iz memorije.
`run`	Pokretanje vanjskog programa ili skripte.

Tablica 12. Opis mogućih podnaredbi u Longview-u.

Podnaredba	Opis
`variable`	Naredba se odnosi na varijablu.
`mesh`	Naredba se odnosi na trodimenzionalni objekt (mesh).
`pdb`	Naredba se odnosi na PDB datoteku.
`node`	Naredba se odnosi na čvor u grafu scene.
`view`	Naredba se odnosi na OpenGL prozor u sučelju.
`ms`	Naredba se odnosi na stvaranje molekularne površine iz učitane PDB molekule.

Tablica 13 . Opis mogućih parametara u Longview-u.

Parametar	Opis
`name`	Vrsta podatka: `string`. Predstavlja ime varijable, objekta, čvora...
`path`	Vrsta podatka: `string`. Putanja do određene datoteke (npr. PDB, mesh,...).

`center`	Vrsta podatka: `string`. Može biti “`true`” ili “`false`” (odnosno bilo što osim “true”). Određuje da li će se objekt centrirati prilikom učitavanja.
`centerType`	Vrsta podatka: `string`. Ako je centerType jednak “`arith`”, centriranje će biti provedeno uz pomoć aritmetičke sredine (atoma u slučaju PDB-a, odnosno točaka modela u slučaju mesh-a), a u slučaju da je centerType jednako “`mass`”, centriranje će biti provedeno u odnosu na težište mase (vrijedi samo kod učitavanja PDB datoteka).
`nodeType`	Vrsta podatka: `string`. Omogućuje odabir vrste novog čvora. Mogućnosti postavke čvora prikazane su u tablici 14.
`nodeParams`	Vrsta podatka: ovisno o tipu čvora. Parametar se može sastojati od jednog broja; riječi koja označava ime varijable; niza brojeva koji označavaju vrijednost komponenti vektora,...
`radiusPath`	Vrsta podatka: `string`. Putanja do datoteke koja sadrži popis radijusa (primjenjuje se kod učitavanja nove PDB datoteke).
`radiusType`	Vrsta podatka: `string`. Postoje dvije mogućnosti: u slučaju kada je vrijednost jednaka “`explicit`”, primjenjuje se eksplicitna vrijednost radijusa za određivanje dimenzija atoma; dok se u protivnome (bilo koja druga vrijednost) koristi ujedinjena ("`united`") vrijednost radijusa.
`value`	Vrsta podataka: ovisan o primjeni. Primjer primjenje: stvaranje nove varijable i dodjeljivanje vrijednosti.
`autoScale`	Vrsta podataka: `double`. Koeficijent koji označava najveću dozvoljenu dimenziju učitanog objekta po bilo kojoj osi. U slučaju da je vrijednost jednaka `0`, objekt neće biti skaliran.

`parent`	Vrsta podatka: `string`. Označava ime roditeljskog čvora, primjenjuje se kod stvaranja novog čvora. Ako je parametar izostavljen, ili čvor na koji se referencira ne postoji, novi čvor će biti dodan izravno na izvorišni (eng. *root*) čvor.

Prilikom kreiranja novih čvorova grafa scene, moguće je svakom čvoru pridodijeliti jednu od funkcija opisanih u poglavlju 4.3. Točni nazivi ovih funkcija za korištenje u skriptnim naredbama prikazani su u tablici 14.

Tablica 14. Imena čvorova korištena u naredbama.

Vrsta čvora	Opis
`nop`	Čvor nema operacije.
`mesh`	Čvor služi za iscrtavanje trodimenzionalnog objekta.
`pdb`	Čvor služi za iscrtavanje molekula učitanih iz PDB datoteka.
`color`	Čvor služi za postavljanje boje.
`lookat`	Čvor služi za postavljanje trenutnog pogleda.
`translation`	Čvor služi za pomicanje po X, Y i Z osi.
`rotationx`	Čvor služi za rotaciju oko X osi.
`rotationy`	Čvor služi za rotaciju oko Y osi.
`rotationz`	Čvor služi za rotaciju oko Z osi.
`rotationeuler`	Čvor služi za rotaciju po Eulerovim kutovima.
`scale`	Čvor služi za skaliranje.
`script`	Čvor služi za izvršavanje skripte na danoj putanji.

U slučaju stvaranja novih varijabli, potrebno je odrediti njihov tip. Točni nazivi tipova varijabli (kako ih treba upotrebljavati u skriptnim naredbama) te njihovi opisi, dani su u tablici 15.

Tablica 15. Imena varijabli korištena u naredbama.

Vrsta varijable	Opis
`char`	Odgovara tipu `char` u C++-u. Veličina: 1 bajt.
`uchar`	Odgovara tipu `unsigned char` u C++-u. Veličina: 1 bajt.
`long`	Odgovara tipu `long in` u C++-u. Veličina: 4 bajta.
`ulong`	Odgovara tipu `unsigned long int` u C++-u. Veličina: 4 bajta.
`ulonglong`	Odgovara tipu `unsigned long long int` u C++-u. Veličina: 8 bajta.
`float`	Odgovara tipu `float` u C++-u. Veličina: 4 bajta.
`double`	Odgovara tipu `double` u C++-u. Veličina: 8 bajta.
`vector3`	Struktura specifična za Longview. Sastoji se od tri realna broja (tipa `double`) koja odgovaraju `x`, `y` i `z` koordinatama. Veličina: 3x8 bajta.
`color`	Struktura specifična za Longview. Sastoji se od četiri cjelobrojna broja (tipa `unsigned char`) koja odgovaraju `r`, `g`, `b` i `a` parametrima boje. Veličina: 4x1 bajt.
`eulervector3`	Struktura specifična za Longview. Sastoji se od tri realna broja (tipa `double`) koja odgovaraju `alfa`, `beta` i `gama` Eulerovim kutovima. Veličina: 3x8 bajta.
`string`	Odgovara tipu `std::string` u C++-u. Dinamički alocira memoriju ovisno o veličini teksta koji joj se dodjeljuje.

Prilikom deklaracije ili mijenjanja objekata tipa mesh, potrebno je odrediti njegov tip. Popis mogućih tipova dan je u tablici 16.

Tablica 16. Opis mogućih tipova mesh-eva u Longview-u.

Tip mesh-a	Opis
`dat`	Odnosi se na modele u *.dat* formatu, specifičnom za ovaj alat.
`grid`	Ovaj tip označava da se učitava trodimenzionalni grid iz *.grid* formata, koji je primjerice rezultat određivanja površine molekule kuglinim funkcijama.
`ms`	Predstavlja molekularnu površinu u formatu kojeg određuje program *MSMS*. Površina se učitava iz dvije datoteke, ekstenzija *.vert* i *.face*.

Vrlo je bitno napomenuti jednu specifičnost Longview-a: sve naredbe izvršavaju se preko varijabli. To, na primjer, znači da kada stvorimo novi čvor za translaciju, zapravo mu predajemo pokazivač (u pozadini sustava, u samoj naredbi to se događa kada navedemo ime varijable) na prethodno definiranu varijablu (stvorenu pomoću naredbe `create`). Na ovaj način korisnik može mijenjati varijablu koja određuje translaciju, a da ne mora direktno mijenjati sam čvor grafa scene. Promjenom varijable, istovremeno se može uočiti i promjena u samom prikazu.

4.4.2. Skripte

Skripte su datoteke građene od upravo opisanih naredbi. One mogu sadržavati opis cijelog prikaza (scene), a osim toga mogu čak i izvršavati razne naredbe te time ostvarivati dinamičnost scene. Primjer takvih akcija je izvršavanje skripte u čvoru grafa scene, u kojoj se mijenja položaj nekog objekta. Objekt će se u realnom vremenu pomicati na prikazu. Skripte je moguće izvršiti na ukupno pet načina:

1. pokretanjem preko izbornika u sučelju
2. pokretanjem preko naredbenog retka
3. pokretanjem iz skripte koja se izvršava
4. stvaranjem skriptnog čvora
5. izvršavanjem početne (eng. *default*) skripte

Kako su prva četiri navedena načina već opisana, potrebno je još objasniti izvršavanje početne skripte. Kako u Longview-u postoji velik broj postavki koje je potrebno izvršiti, nije vrlo jednostavno u kratkom vremenu prikazati model ili molekulu. U svrhu pojednostavljivanja postavljanja potrebnih parametara, alat omogućuje pokretanje početne skripte – skripte koja se izvršava prilikom svakog pokretanja Longview-a. Time se, na primjer, mogu postaviti sve potrebne varijable i postavke pogleda, te jednostavno učitati PDB datoteke. Početnu skriptu moguće je mijenjati iz samog sučelja, ili ručnom promjenom tekstualne datoteke.

4.4.3. Upotreba naredbi skriptnog jezika

Stvaranje varijabli

Svi tipovi varijabli stvaraju se na isti način, koji je pokazan sljedećim primjerom:

```
create variable name={imeVarijable1} type={char} value={5}
```

Za razliku od običnih skalarnih tipova varijabli (`char, uchar, float`,...), vektorski tipovi varijabli (`vector3, color` i `eulervector3`) deklariraju se tako da se vrijednosti svake dimenzije vektora međusobno odvoji znakom razmaka:

```
create variable name={imeVarijable2} type={vector3} value={1 2
3}
```

Dodatna mogućnost je korištenje heksadecimalnih brojeva prilikom deklaracije varijabli tipa `ulonglong`. Ako parametar `value` počinje znakovima "`0x`", tada broj koji slijedi predstavlja heksadecimalni zapis:

```
create variable name={imeVarijable3} type={ulonglong}
value={0x0000000000001105}
```

Stvaranje mesh-eva

Primjer učitavanja mesh-a koji će biti centriran i skaliran tako da mu je maksimalna veličina po svim osima jednaka 1.0.

```
create mesh path={data/models/sphere.dat} name={sphere}
type={dat} center={true} autoScale={1.0}
```

Mesh-evi tipa `ms` učitavaju se na jednak način, dok se kod mesh-eva tipa `grid` trebaju navesti i dodatne vrijednosti u `value` parametru:

```
create mesh name={mesh1} path={data/grid/testGrid.grid}
type={grid} center={false} autoScale={0.00} value={false 0.00}
```

Prva vrijednost može biti "`true`" ili "`false`", a označava da li želimo da se referentna isovrijednost odredi automatski. Ako je postavljena u "`true`", druga vrijednost se zanemaruje. U slučaju da je "`false`", druga vrijednost će se uzeti kao referentna isovrijednost za prikaz površine.

Stvaranje PDB-ova

Sljedeći primjer pokazuje kako se može učitati nova PDB datoteka. Molekula će biti centrirana prema masi. Ako je parametar `center` postavljen u "`false`", parametar `centerType` će biti zanemaren.

```
create pdb path={data/pdb/3hfl.pdb} name={complex}
radiusPath={data/system/atmtypenumbers} radiusType={united}
center={true} centerType={mass}
```

Stvaranje molekularne površine

Površina molekule može se odrediti korištenjem dolje navedene naredbe. Nakon stvaranja nove površine, u Longview okruženju ne mogu se vidjeti nikakvi rezultati, jer se površina pohranjuje u datoteke na disku. Parametar `path` određuje putanju do tih datoteka (generiraju se dvije datoteke: `<path>.vert` i `<path>.face`). Parametar `value` određuje ime varijable koja sadrži mode sa željenim postavkama, kako je to opisano u tablici 10. Parametar `name` određuje ime PDB-a, koji prethodno mora biti

učitan, a iz kojeg ćemo generirati površinu.

```
create ms name={complex} path={data/temp/msComplex}
value={nodeModeComplex}
```

Stvaranje čvorova grafa scene

Primjer prikazuje način stvaranja čvorova grafa scene. U primjeru su stvorena dva čvora, koja u međusobno povezana: čvor `nodeLook1` je izvorišni čvor (eng. *root*), a istovremeno je i roditelj čvora `nodeTranslateReceptor` (eng. *parent*). Pri tome svi objekti na koje se referencira moraju prethodno biti deklarirani (u ovom primjeru: `vecLook1, vecPoint1, vecUp1` i `translationReceptor`).

```
create node name={nodeLook1} nodeType={lookat}
nodeParams={vecLook1 vecPoint1 vecUp1}

create node name={nodeTranslateReceptor} nodeType={translation}
nodeParams={translationReceptor} parent={nodeLook1}
```

Stvaranje novog pogleda

Longview omogućuje prikaz s više pogleda istovremeno. Sljedeći primjer pokazuje kako se može dodati novi pogled u sučelje alata.

```
create view name={view1} nodeParams={nodeLook1}
```

Moguće je stvoriti najviše četiri različita pogleda. Parametar `nodeParams` određuje koji čvor grafa scene će biti referentni izvorišni čvor za iscrtavanje pogleda.

Mijenjanje postavki stvorenim objektima

Postavke stvorenih objekata moguće je mijenjati korištenjem naredbe `change`. Prvi parametar nakon naredbe change treba odgovarati imenu objekta koji se želi promijeniti, a nakon njega slijede parametri (isti kao i kod stvaranja tog objekta). Primjer pokazuje način upotrebe ove naredbe.

```
change imeVarijable1 value={10}
```

Matematičke operacije

Naredba `math` ima sličnu funkciju kao i `change`, uz tu razliku što se kod ove naredbe izrazi zadani u `value` parametru evaluiraju kao matematički izrazi. Naredba `math` može se primijeniti samo na skalarne varijable (dakle, na sve varijable osim `vector3`, `color` i `eulervector3`). Primjer upotrebe `math` naredbe:

```
math name={imeVarijable1} value={:imeVarijable1 +
:imeVarijable2}
```

Parametrom `name` određuje se varijabla u koju će se rezultat pohraniti, dok se u parametru `value` nalazi izraz kojeg je potrebno evaluirati.

Pokretanje izvršnih programa ili skripti

Longview omogućuje pokretanje vanjskih izvršnih programa (u slučaju kada je parametar `type` postavljen na vrijednost "`exe`"), te izvršavanje vanjskih skripti (kada je parametar `type` postavljen na vrijednost "`script`"). U slučaju skripte, parametar `value` određuje putanju do skripte, dok u slučaju izvršnih programa parametar `value` određuje putanju do tog programa kao i sve potrebne parametre za njegovo izvršavanje. Primjer upotrebe:

```
run type={script} value={data/script/testSkripta.vsc}
```

Uklanjanje objekata

Naredbom `remove` stvoreni objekti mogu se ukloniti iz radne okoline, i to tako da se nakon imena naredbe navede ime objekta kojeg želimo ukloniti. Primjena naredbe jednaka je za sve vrste objekata. Primjer upotrebe:

```
remove imeVarijable1
```

4.5. Sučelje

U Longview-u, sučelje je samo svojevrsna "maska" u odnosu na funkcionalnost sustava. Gotovo sve mogućnosti dostupne iz sučelja mogu se izvršiti ručnim upisivanjem skriptnih naredbi u naredbenu liniju, ili učitavanjem skripte. Štoviše, gotovo sve akcije sučelja se i zvršavaju na sličan način – ovisno o tome koji se događaj ostvario te koji su parametri odabrani, generira se upravo tražena naredba skriptnog jezika, koja se neposredno nakon toga izvršava. Ovakav način organizacije upravljanja alatom Longview pruža neke zanimljive i vrlo korisne mogućnosti.

Prvo, kako se sve izvršene skriptne naredbe mogu pohraniti u memoriju, pa tako i naredbe generirane od strane sučelja, to omogućava funkciju spremanja cijele radne okoline i nastavka rada u istoj okolini nakon što je alat ponovno pokrenut. Drugo, moguće je definirati *makro naredbe*, koje predstavljaju isječke koda skriptnog jezika nastalih, na primjer, postavljanjem ili mijenjanjem opcija iz sučelja. Ovakve makro naredbe mogu se kasnije pokretati, te tako olakšati izvršavanje nekog repetitivnog niza akcija.

4.5.1. Izvedba sučelja

Sučelje je, kao i ostali dijelovi alata Longview, u potpunosti napisano u C++-u. Pri tome je korištena biblioteka *FLTK*. FLTK je skraćenica od eng. *Fast Light Toolkit*, a predstavlja višeplatformski C++ alat za dizajn grafičkog korisničkog sučelja (eng. *Graphical User Interface*, *GUI*) [7]. Trenutno je podržana na *Unix/Linux (X11)*, *Windows* i *MacOS X* platformama. Ova biblioteka pruža modernu funkcionalnost sučelja s podrškom za 3D grafiku preko OpenGL-a, te je uz to dovoljno mala i modularna da ju je moguće statički povezati (eng. *statical link*) s ostatkom sustava. Statičko povezivanje s bibliotekom znači da se cijela biblioteka kompajlira (eng. *compile*) zajedno s Longview-om u jedan izvršni program. Longview koristi inačicu 1.3 FLTK biblioteke. Iako postoji pomoćni alat za vizualni dizajn grafičkog sučelja koje koristi ovu biblioteku (*Fast Light User Interface Designer, FLUID*), u ovom radu svi dijelovi sučelja ručno su postavljeni na željene položaje. Glavni razlog tome je način organizacije i strukture kôda koji FLUID generira.

Programski, sučelje je izvedeno kao zasebna klasa, nazvana `VmolGui`, te je time kao cjelina odvojeno od ostalih dijelova sustava. Kako može postojati samo jedno korisničko sučelje u jednoj instanci programa, ova klasa realizirana je kao *singleton*. Time je omogućeno pristupanje pojedinim resursima sučelja izvana (na primjer iz drugih klasa). `VmolGui` intenzivno komunicira s klasom `VmolCommand`, i to prilikom izvršavanja pojedinih naredbi iz sučelja ili prikaza stanja radne okoline. Najbitnije javne funkcije sučelja su:

- funkcija za stvaranje i postavljanje sučelja

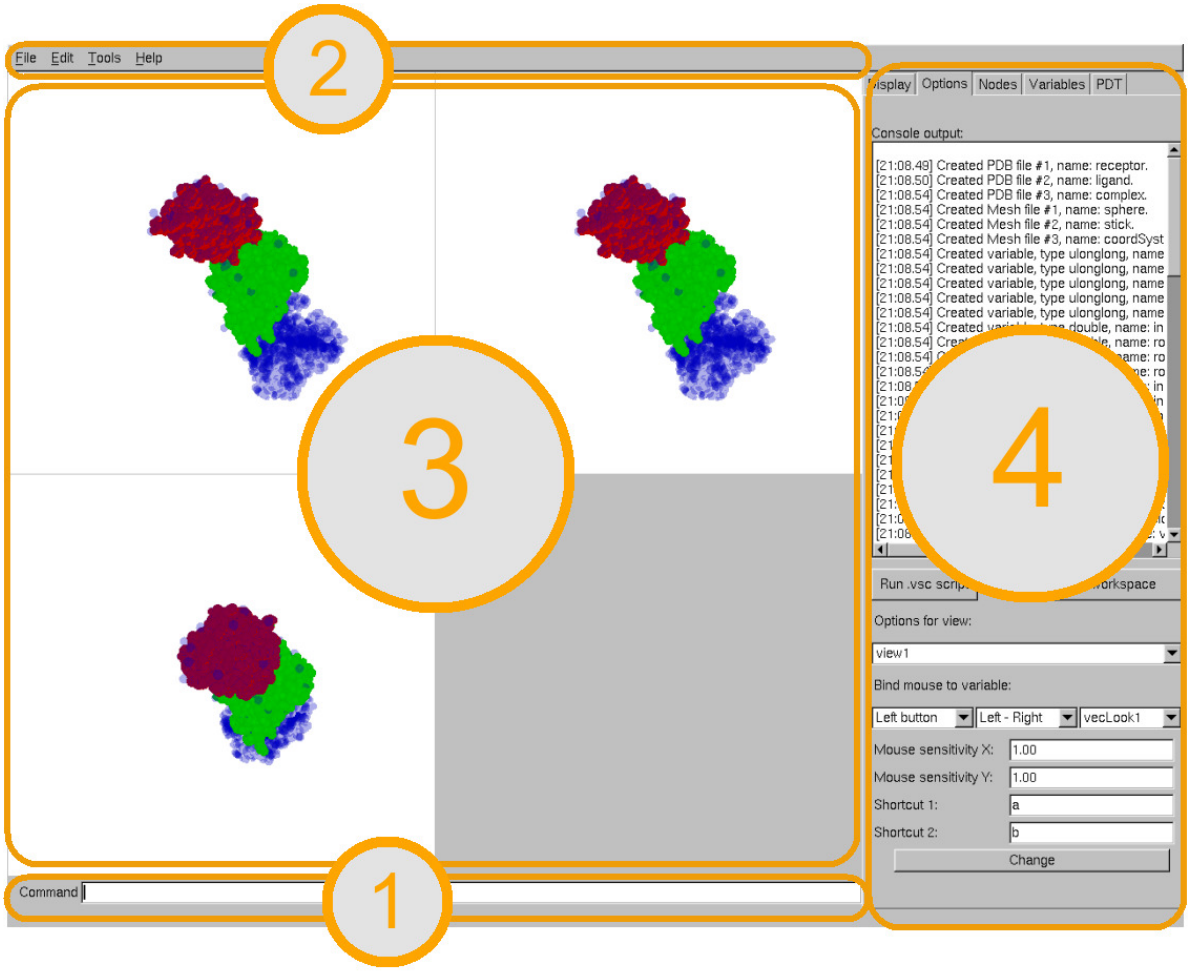
```
bool createGui();
```

- funkcija za osvježavanje vrijednosti prikazanih u sučelju (popisi varijabli, pogleda, akcija miša,...)

```
void refreshGui();
```

Postoje i brojne vrlo važne privatne funkcije, a većina ih je zadužena za izvršavanje naredbi prilikom pritiska na gumb ili neku drugu kontrolu u sučelju (eng. *callback functions*). Osim funkcija, u privatnom dijelu klase nalaze se i pokazivači na svaku kontrolu prikazanu u sučelju. Time je omogućen pristup svakoj kontroli, kao i mijenjanje njezinih svojstava iz `VmolGui` klase.

4.5.2. Izgled sučelja



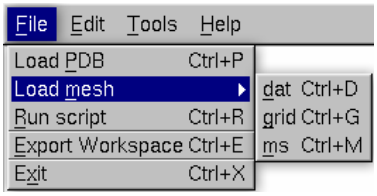
Slika 18. Sučelje Longview-a s označene četiri bitne cjeline.

Slika 18 prikazuje sučelje alata Longview. Na slici su narandžastom bojom označene četiri bitne cjeline. To su:

- naredbeni redak za zadavanje naredbi skriptnog jezika
- padajući izbornik
- područje za prikaz pogleda i pregled modela
- glavni izbornik s najbitnijim opcijama

Padajući izbornici

Slika 19 prikazuje padajući izbornik *File*. U njemu se mogu pronaći opcije za učitavanje PDB datoteka (*Load PDB* izbornik), učitavanje mesh-eva sva tri podržana tipa (izbornik *Load mesh*), izvršavanje skripti (izbornik *Run script*), pohranjivanje radne okoline u datoteku (izbornik *Export Workspace*), te izlaz iz programa (izbornik *Exit*).



Slika 19. Izbornik File.

Za razliku od učitavanja PDB i mesh datoteka preko skriptnih naredbi ili glavnog izbornika, kod ovakvog načina učitavanja koriste se pretpostavljene (eng. *default*) vrijednosti. Pretpostavljene vrijednosti prilikom učitavanja PDB datoteka su: ime oblika *pdbN* gdje je *N* ukupan broj već učitanih PDB-ova, putanja do datoteke s radijusima jednaka "`data/system/atmtypenumbers`", korištenje ujedinjenog radijusa te isključeno centriranje molekula. Kod učitavanja novog mesh-a pretpostavljene vrijednosti su: ime oblika *meshN* gdje je *N* ukupan broj već učitanih mesh-eva, centriranje modela isključeno, isključeno skaliranje, te uključeno automatsko određivanje referentne isovrijednosti za gridove. Također, prilikom učitavanja novih PDB i mesh datoteka, za svaku od njih automatski se stvori novi mode, te jedan čvor. Čvor se postavlja kao dijete zadnjem čvoru dodanom u prikaz.

Slika 20 prikazuje izgled izbornika *Edit*. U trenutku pisanja ovog rada, izbornik sadrži samo jednu opciju, a to je mogućnost postavljanja trenutne radne okoline kao pretpostavljenog (početnog) okruženja, koje se učitava prilikom svakog pokretanja alata Longview.

Slika 20. Izbornik Edit.

Na slici 21 prikazan je izbornik *Tools*. On omogućuje pokretanje postupka snimanja makro naredbi (*Start recording macro*), te zaustavljanje tog postupka i pohranjivanje rezultata na željenu putanju (*Stop recording macro*).

Slika 21. Izbornik Tools.

Slika 22 prikazuje izbornik Help. On sadrži opciju About, gdje se mogu pronaći informacije o autoru, te kontakt adresa elektroničke pošte.

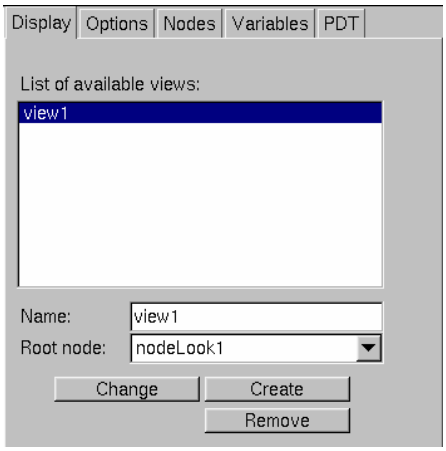
Slika 22. Izbornik Help.

Glavni izbornik

Glavni izbornik sastoji se od pet tabova (eng. *tab*), od kojih svaki sadrži specifičnu grupu opcija. Tabovi su redom: *Display*, *Options*, *Nodes*, *Variables* i *PDT*.

Tab *Display*:

Tab *Display*, prikazan slikom 23, sadrži mogućnosti prilagodbe prikaza.



Slika 23. Glavni izbornik, tab Display.

Izbornik prikazuje popis pogleda stvorenih u okolini. Pojedini pogled može biti stvoren na način kako je to opisano u poglavlju 4.4, ili direktno korištenjem ovog izbornika. Odabirom nekog od pogleda iz popisa, istovremeno se prikazuju njegovi parametri u kontrolama izbornika. Konkretno, parametri pogleda obuhvaćaju njegovo ime te izvorišni čvor za prikaz. Ove parametre moguće je i mijenjati, a promjena se dogodi u trenutku kada korisnik pritisne gumb "*Change*". Pri tome će se promijeniti parametri onog pogleda koji je trenutno odabran na popisu, neovisno o ručno upisanom imenu. Razlog tome je mogućnost promjene imena željenog pogleda. Odabirom gumba "*Create*" može se stvoriti, a gumbom "*Remove*" obrisati odabrani pogled.

Tab *Options*:

Izgled taba *Options* prikazan je slikom 24. Na vrhu izbornika, pod naslovom "*Console output*", može se pronaći tekstualni ispis rezultata izvođenja naredbi skriptnog jezika. Neposredno ispod tog ispisa nalaze se dva gumba:

- "*Run .vsc script*" koji služi za pokretanje skripte na određenoj putanji
- "*Export workspace*" koji služi za pohranjivanje trenutnog stanja okoline na određenu putanju

Preostale mogućnosti ovog izbornika odnose se na podešavanje postavki miša.

Padajućim izbornikom pod naslovom "*Options for view*" određuje se pogled u prikazu na kojeg se odnose odabrane postavke. Ispod naslova "*Bind mouse to variable*" nalaze se tri padajuća izbornika, pomoću kojih se radnje miša povezuju sa željenim akcijama. Radnja miša je predstavljena u obliku pomaka miša u određenom smjeru. Osim pomaka miša, za izvršavanje određene akcije potrebno je i pritisnuti određenu tipku miša, ili kombinaciju tipke miša s tipkom na tipkovnici. Zatim se svakoj odabranoj kombinaciji pritisaka tipki i pomaka miša pridodjeljuje željena postojeća varijabla. Ovo **svojstvo** je, u usporedbi s ostalim alatima namijenjenim vizualizaciji, **specifično za Longview**, a korisniku omogućuje **interaktivno podešavanje gotovo svih postavki okoline i pogleda**. Razlog tome je što se gotovo sve skriptne naredbe ostvaruju preko pokazivača na varijable. Time izravna promjena pojedine varijable izravno utječe i na svojstva prikaza. To znači da je, na primjer, pomakom miša u kombinaciji s pritisnutom lijevom tipkom moguće kontrolirati promjene kuta gledišta, položaj pojedinih objekata na sceni, boju objekata, pa sve do *mode* postavki prikaza. Na kraju, mogućnosti su zapravo neograničene, a ovise samo o korisnikovim postavkama okoline, tj. primjeni tih varijabli. Kako su neke varijable skalarne, a neke vektorske, njihove vrijednosti se mijenjanju prema sljedećem principu:

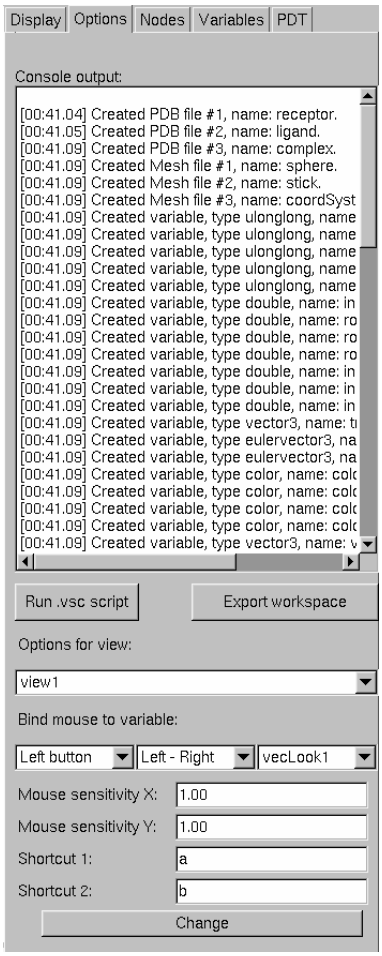
- skalarne varijable
    - jednodimenzionalne, vrijednost im se mijenja točno kako je to zadano odabranom radnjom miša
- vektorske varijable
    - trodimenzionalne (vector3, eulervector3)
        - vrijednost prve dimenzije povezuje se s pomakom miša u smjeru lijevo – desno
        - vrijednost druge dimenzije povezuje se s pomakom miša u smjeru gore – dolje
        - vrijednost treće dimenzije povezuje se s pomakom miša u kombinaciji s prvom pritisnutom tipkom tipkovnice
    - četverodimenzionalne (color)
        - vrijednost prve tri dimenzije mijenja se kao i kod trodimenzionalnih varijabli

- vrijednost četvrte dimenzije povezuje se s pomakom miša u kombinaciji s drugom pritisnutom tipkom tipkovnice

Spomenutim padajućim izbornicima svrha je:

1. lijevi izbornik – odabir pritisnute tipke miša na koju se odnosi akcija (lijeva, srednja, desna, te kombinacija svake od njih s dvije moguće tipke s tipkovnice)
2. srednji izbornik – odabir smjera pomaka miša (lijevo – desno ili gore – dolje)
3. desni izbornik – odabir varijable na koju se akcija miša primjenjuje

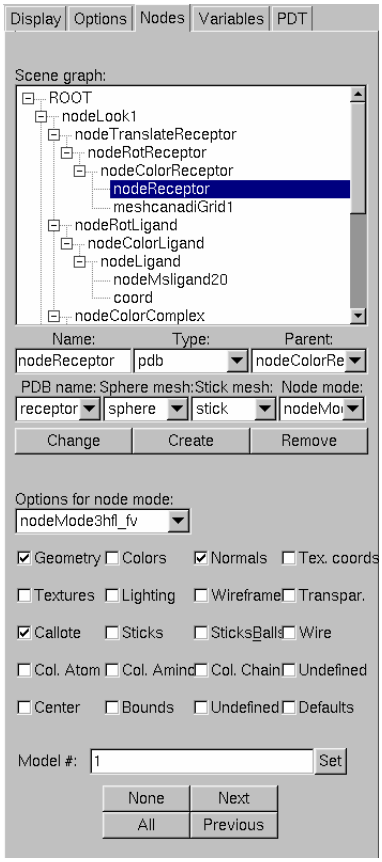
U tekstualnim okvirima označenim s "*Mouse sensitivity X*" odnosno "*Mouse sensitivity Y*" upisuju se željene (realne) vrijednosti osjetljivosti miša za pomak u vodoravnom (*X*) smjeru, odnosno horizontalnom (*Y*) smjeru, dok tekstualni okviri pod naslovom "*Shortcut 1*" i "*Shortcut 2*" služe za odabir tipki tipkovnice koje će biti korištene u kombinaciji s tipkama miša. Gumb "Change" služi za promjenu trenutno odabranih postavki.



Slika 24. Glavni izbornik, tab Options.

Tab *Nodes*:

Tab *Nodes* sadrži sve potrebne mogućnosti za stvaranje, mijenjanje ili brisanje postojećih čvorova grafa scene. Osim toga, na njemu se nalaze i kontrole za interaktivno mijenjanje postavki prikaza mesh i PDB modela (mode). Prikaz Nodes izbornika dan je na slici 25.



Slika 25. Glavni izbornik, tab Nodes.

Na samom vrhu izbornika ispod naslova "*Scene graph*" nalazi se kontrola, koja prikazuje strukturu oblika stabla. To stablo predstavlja postojeći graf scene, a svaki redak predstavlja naziv jednog čvora grafa. Prikaz stabla je ostvaren *uvlačenjem* naziva čvorova djece u odnosu na naziv njegovog roditelja. Tako se, na primjer, na slici 25 može vidjeti kako čvorovi "*nodeMsligand20*" i "*coord*" imaju istog roditelja – "*nodeLigand*", čiji je roditelj "*nodeColorLigand*", i tako dalje. Odabirom imena pojedinog čvora, u padajućim izbornicima ispod prikaza stabla prikazuju se njihove postavke. Moguće ih je mijenjati, nakon čega je potrebno pritisnuti gumb "*Change*", ili brisati, gumbom "*Remove*". U oba slučaja, čvor na kojeg se takva radnja odnosi mora biti odabran u prikazu stabla. Pritiskom na gumb "*Create*" moguće je stvoriti novi čvor s odabranim postavkama (naravno, čvor mora imati različito ime od svih već

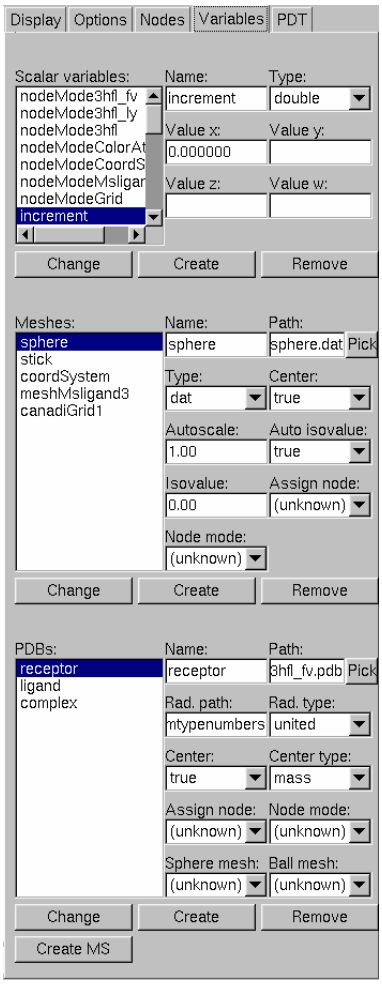
postojećih čvorova). Nazivi padajućih izbornika, te rezultati odabira pojedinih postavki isti su kao i kod skriptnih naredbi vezanih uz čvorove grafa scene. Dodatne informacije o njima mogu se pronaći u poglavlju 4.4.

Padajući izbornik označen s "*Options for node mode*" omogućuje odabir jedne od postojećih varijabli tipa *ulonglong.* Neposredno nakon odabira varijable, uključuju se ili isključuju oznake u kućicama na drugoj polovici izbornika, a osim toga mijenja se i vrijednost okvira označenog s "*Model #*". Ove kućice predstavljaju pojedine bitove odabrane varijable, čija funkcija je opisana u tablici 10, poglavlje 4.3. Broj upisan u okvir "*Model #"* predstavlja indeks modela koji se prikazuje (odnosi se samo na PDB datoteke s više modela).

Pritiskom na određenu kućicu istovremeno se mijenja vrijednost odgovarajućeg bita podatka. Ako se želi promijeniti vrijednost indeksa prikazanog modela, potrebno je pritisnuti gumb "*Set*". Gumbi "*Next*" i "*Previous*" povećavaju, odnosno smanjuju vrijednost indeksa prikazanog u okviru "*Model #*", dok gumb "*None*" služi za upis vrijednosti koja neće prikazati niti jedan model (vrijednost *0*), a gumb "*All*" upisuje vrijednost koja će prikazati sve trenutno učitane modele (vrijednost $(2^{16}-1)$).

Tab *Variables*:

Ovaj tab sadrži sve podatke o varijablama (skalarnim i vektorskim), te mesh i PDB modelima, te je na taj način i podijeljen u tri logičke cjeline. Kao i na prethodnim tabovima, pritiskom na gumbe "Create" i "Change" može se stvoriti ili promijeniti, te gumbom "Remove" obrisati odabrana varijabla, mesh ili PDB. Slika 26 prikazuje izbornik "Variables".



Slika 26 . Glavni izbornik, tab Variables.

Svaka cjelina sastoji se od popisa postojećih varijabli (odnosno mesh i PDB modela) na lijevoj strani taba, te tekstualnih okvira i padajućih izbornika koji prikazuju svojstva odabrane varijable (odnosno mesh ili PDB modela) s desne strane taba.

Prva cjelina odnosi se na varijable. Okvir "*Name*" prikazuje ime, a izbornik "*Type*" tip odabrane varijable, odnosno varijable koja će biti stvorena. Kako varijable mogu imati i do četiri dimenzije, vrijednosti svake dimenzije prikazuju se, odnosno mogu se upisivati, u tekstualne okvire označene s:

- "*Value x*" – prva dimenzija
- "*Value y*" – druga dimenzija
- "*Value z*" – treća dimenzija
- "*Value w*" – četvrta dimenzija

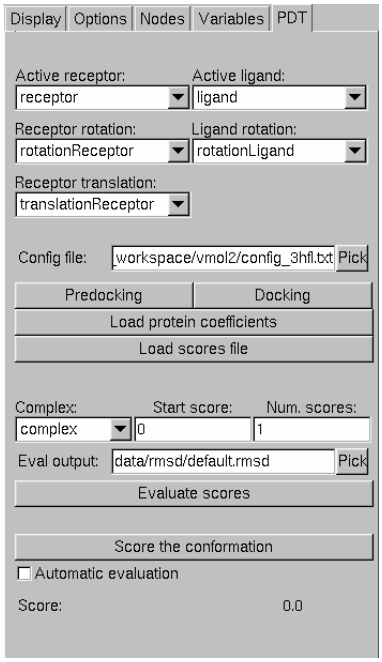
Kada bi se ove vrijednosti spojile u jedan tekstualni izraz (odvojene razmacima), dobio bi se oblik *value* parametra kod stvaranja ili mijenjanja varijabli putem skriptnih naredbi.

Druga cjelina odnosi se na mesh modele. Ovdje se nalaze mogućnosti pregleda ili promjene svih parametara pojedinog mesh-a, koje su nabrojene i opisane u poglavlju 4.4. Dodatne mogućnosti koje nudi ovaj izbornik vezane su uz automatizirano dodavanje novostvorenog mesha određenom čvoru grafa scene, sa željenim mode-om. Padajućim izbornikom "*Assign node*" odabire se koji će čvor biti roditeljski čvor ovom modelu, dok su parametri modela zadani mode-om odabranim u padajućem izborniku "*Node mode*". Ukoliko je vrijednost nekog od ova dva parametra jednaka "*(unknown)*", novi mesh neće biti dodan u graf scene.

Treća cjelina se odnosi na molekule učitane iz PDB datoteka. Analogno drugoj cjelini, ovdje se nalaze sve postavke o kojima ovisi učitani model, te opcije za odabir roditeljskog čvora i željenog mode-a.

Tab *PDT*:

Na ovom tabu nalaze se sve opcije vezane uz proces prianjanja proteina, a slika 27. prikazuje izgled ovog taba.



Slika 27. Glavni izbornik, tab PDT.

Preko padajućih izbornika "*Active receptor*" i "*Active ligand*" odabiru se molekule učitane iz PDB datoteka koje će biti korištene prilikom evaluacije rezultata ili ocjenjivanja konformacija. Padajući izbornici "*Receptor rotation*" i "*Ligand rotation*" omogućuju odabir varijabli tipa `eulervector3`, a predstavljaju parametre rotacije, dok izbornik "*Receptor translation*" omogućuje odabir varijabli tipa `vector3`, a predstavlja parametar translacije prilikom ocjenjivanja konformacija. Pojedinačno ocjenjivanje konformacija može se izvršiti pritiskom na gumb "*Score the conformation*". Isto tako, ocjenjivanje je moguće raditi i automatski prilikom pomicanja ili rotacije molekula, označavanjem kućice "*Automatic evaluation*". Rezultat ocjenjivanja prikazan je pokraj oznake "*Score*".

U tekstualnom okviru pod oznakom "*Config file*" nalazi se putanja do konfiguracijske datoteke korištene u procesu prianjanja proteina. Kako se taj proces sastoji od dvije glavne cjeline (*predocking* i *docking*), tako postoje i dva odgovarajuća gumba u ovom izborniku. Gumb "*Predocking*" započinje proces računanja koeficijenata sfernih harmonika unutarnje i vanjske ljuske proteina, dok se proces prianjanja koji koristi te koeficijente pokreće pritiskom na gumb "*Docking*". Pri tome se koriste postavke iz odabrane konfiguracijske datoteke. Izračunati koeficijenti također se primjenjuju i prilikom ocjenjivanja konformacija. Rezultati procesa prianjanja proteina pohranjuju se u datoteku u obliku rang liste konformacija. Te rezultate moguće je učitati pritiskom na gumb "Load scores file". Svrha ove opcije je proračun *točnosti* pojedine konformacije, koja se dobije u odnosu na *točan* izgled željenog kompleksa (kristalografske strukture). Mjera točnosti je vrijednost *RMSD*, opisana u poglavlju 2.6. Točan kompleks u sučelju se odabire unutar padajućeg izbornika "*Complex*". Postupak računanja točnosti pokreće se pritiskom na gumb "*Evaluate scores*", a rezultati se upisuju u datoteku na putanji određenoj tekstualnim okvirom pod oznakom "*Eval output*". Okvir "Start score" određuje početnu konformaciju iz ulazne datoteke, dok okvir "Num. scores" određuje koliko konformacija se želi ocijeniti.

4.6. Pogreške

Pogreške u Longview-u mogu nastati izvršavanjem naredbi skriptnog jezika ili neočekivanim događajima koji su rezultat izvođenja pojedinih funkcija samog alata ili sustava za prianjanje proteina. Prilikom izvršavanja naredbi, pogreške mogu nastati kao posljedica:

- neispravnih vrijednosti parametara
    - vrijednost parametra zadana je brojčano, a trebala bi biti zadana imenom objekta (ili obrnuto)
    - ime objekta predanog kao parametar ne postoji, ili je imenovani objekt drukčijeg tipa od pretpostavljenog
    - putanja do datoteke ne može biti otvorena
- uklanjanjem stvorenog objekta, koji je povezan s nekim drugim objektom
    - primjer: uklanjanje mesh-a koji se poziva iz čvora grafa scene, dok isti čvor još postoji

U drugom navedenom slučaju, pogreške mogu nastati izvođenjem funkcija koje nisu povezane sa skriptnim naredbama. Primjeri takvih funkcija su:

- učitavanje raznih datoteka
- pokretanje procesa prianjanja proteina
- matematičke funkcije
- ...

Longview provjerava povratne vrijednosti svake izvršene funkcije, te u slučaju da je ona neispravna, dodaje određenu poruku u zapisnik, a u kritičnim slučajima prekida se i izvršavanje samog alata.

Dodatni mogući izvori pogrešaka su preveliki memorijski zahtjevi. Prilikom svake promjene zauzeća memorije Longview prati ispravnost zadane operacije. U slučaju nemogućnosti njenog izvršavanja, alat dodaje objašnjenje pogreške u zapisnik, te prekida svoje daljnje izvođenje.

4.6.1. Procesiranje pogrešaka

U alatu Longview definirana je posebna klasa koja služi za bilježenje poruka u zapisnik (eng. *log*), te vođenje brige o nastalim pogreškama. Ova klasa zove se *Log*, a dizajnirana je kao *singleton*. Sadrži dvije vrlo bitne funkcije, koje obavljaju upravo opisane radnje:

- funkcija za dodavanje poruka u zapisnik

```
int writeToLog(const char *logMessage, ...);
```

- funkcija za vođenje brige o pogreškama

```
int errorReport(std::string errorSeverity, unsigned int
      errorNumber, const char *errorMessage, ...);
```

Pri tome parametri `logMessage` i `errorMessage` predstavljaju tekstualnu poruku koja će biti dodana u zapisnik, `errorSeverity` *težinu* nastale pogreške, a `errorNumber` oznaku vrste pogreške koja se dogodila. Težina poruke najčešće je "`WARNING`" ili "`ERROR`", a može biti i proizvoljno definirana. Njezina uloga je označavanje poruke o pogrešci u zapisniku radi preglednosti. Druga funkcija težine poruke je prekid izvršavanja programa u slučaju da se dogodila pogreška oblika "`ERROR`". Tablica 17 opisuje pojedinu vrstu pogreške definiranu `errorNumber` parametrom.

Tablica 17 . Opisi definiranih vrsta pogrešaka.

errorNumber	Naziv	Opis
1	ERR_MEMORY	Problemi vezani uz memoriju.
2	ERR_OPENING_FILE	Nemoguće otvaranje datoteke.
3	ERR_COMMAND_WRONG_PARAM	Pogrešno definiran parametar skriptne naredbe.
4	ERR_COMMAND_NOT_IMPLEMENTED	Skriptna naredba još nije implementirana.
5	ERR_COMMAND_WRONG_VAR	Pogrešno definiran tip varijable u skriptnoj naredbi.
6	ERR_COMMAND_CANT_EXECUTE	Skriptnu naredbu je nemoguće izvršiti.
7	ERR_RETURNED_FUNCTION	Pozvana funkcija nije uspješno izvršena.

4.6.2. Otpornost na pogreške

Svaka funkcija alata Longview izvedena je tako, da ukoliko izvršava dio koda osjetljiv na pojavu pogreške, vraća vrijednost tipa `int`. Povratna vrijednost jednaka je `0` ukoliko je funkcija uspješno izvršena, a različita od `0` u protivnome. Odabran je ovakav način provjera pogrešaka, zbog mogućnosti određivanja vrste pogrešaka iz povratne vrijednosti funkcije (mogla se dogoditi jedna od više različitih mogućih pogrešaka, dok je moguć samo jedan slučaj ispravnog izvršavanja funkcije).

Osim otpornosti na pogreške čiji je izvor unutar samog alata, Longview mora biti otporan i na pogreške u ulaznim podatcima. Kako su ulazni podatci predstavljeni u obliku datoteka koje sadrže informacije o pojedinom proteinu, modelu, skripti, itd., otkrivanje/ispravljanje pogrešaka događa se u trenutku njihovog učitavanja.

PDB datoteke

Zapis PDB datoteke je tekstualnog oblika s točno određenim značenjem svakog znaka u pojedinom retku. Kako se podatci iz datoteke učitavaju redak po redak, pogreška može nastati ukoliko neki redak ima broj znakova manji od 80 (veličina retka definirana PDB formatom). Ukoliko je broj znakova manji od 80, redak se automatski proširuje prazninama do te veličine. Kako se svi parametri očitavaju iz točno određenih pozicija retka, te iako vrijednosti na tim pozicijama zbog proširivanja redaka možda neće biti ispravne, na ovaj način spriječit će se nestabilnost alata i omogućiti njegov rad i pod manje povoljnim uvjetima.

Datoteke za opis trodimenzionalnih modela (mesh-eva)

Ovakve vrste datoteka generirane su od strane drugih programa, te se njihova ispravnost pretpostavlja (do određene razine). Sama struktura ovih datoteka se ne provjerava. Otpornost na pogreške kod učitavanja ovih datoteka javlja se u obliku pogrešno definiranog broja vrijednosti za učitavanje (npr. točaka) na početku datoteke. Ova vrijednost se zanemaruje, a kao točna uzima se ona dobivena samim učitavanjem.

Skripte

Također su dane tekstualnim zapisom. Skriptne naredbe učitavaju se redak po redak iz datoteke, pri čemu ne može nastati pogreška. Jedini mogući trenutak nastanka pogreške je prilikom izvođenja učitane naredbe. Otpornost na ovakav oblik pogreške opisana je u prethodbom poglavlju.

5. Prikaz testiranja

U ovom poglavlju biti će prikazano nekoliko testova provedenih nad sustavom kako bi se potvrdila njegova ispravnost. Svaki test sastojati će se od dva dijela: skripte koja se pokreće i rezultata u obliku slike alata.

5.1. Učitavanje i prikaz PDB datoteka

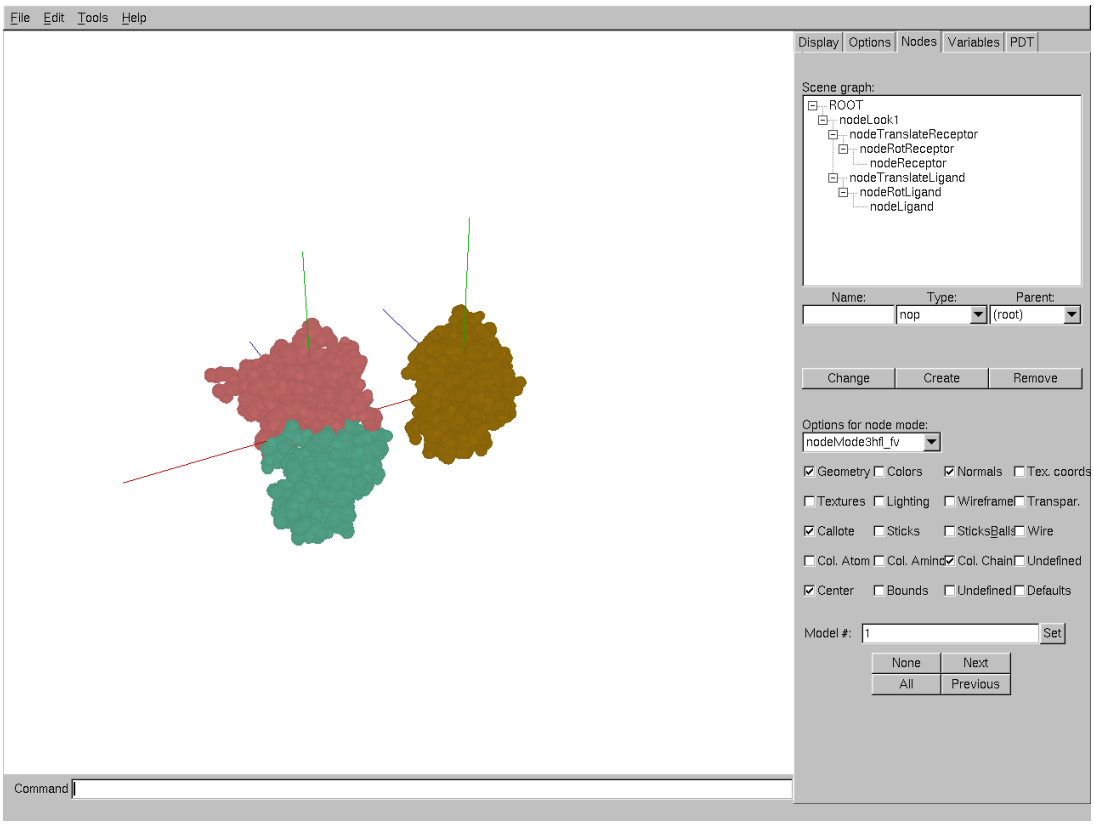
Ovaj test prikazuje postupak učitavanja i prikaza dvije molekule sadržane unutar PDB datoteka. Prije samog prikaza, molekule su međusobno razmaknute po *X*-osi, dok, iako to dolje navedena skripta omogućava, rotacija nije korištena u prikazu. Molekule su obojane po lancima.

Skripta:

```
%%%%% UCITAVANJE PDB-ova %%%%%
create pdb path={data/pdb/3hfl_fv.pdb} name={receptor}
radiusPath={data/system/atmtypenumbers} radiusType={united} center={true}
centerType={mass}
create pdb path={data/pdb/3hfl_ly.pdb} name={ligand}
radiusPath={data/system/atmtypenumbers} radiusType={united} center={true}
centerType={mass}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% UCITAVANJE MESH-eva %%%%%
create mesh path={data/models/sphere.dat} name={sphere} type={dat}
center={true} autoScale={1.0}
create mesh path={data/models/stick.dat} name={stick} type={dat}
center={true} autoScale={1.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VARIJABLI %%%%%
create variable={nodeMode3hfl_fv} type={ulonglong}
value={0x0001000000014105}
```

```
create variable={nodeMode3hfl_ly} type={ulonglong}
value={0x0001000000014105}

create variable name={translationReceptor} type={vector3} value={25.0 0.0
0.0}
create variable name={rotationReceptor} type={eulervector3} value={0.0
0.0 0.0}
create variable name={translationLigand} type={vector3} value={-20.0 0.0
0.0}
create variable name={rotationLigand} type={eulervector3} value={0.0 0.0
0.0}

create variable name={vecLook1} type={vector3} value={0.0 0.0 -180.0}
create variable name={vecPoint1} type={vector3} value={0.0 0.0 0.0}
create variable name={vecUp1} type={vector3} value={0.0 1.0 0.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE NODE-ova %%%%%
create node name={nodeLook1} nodeType={lookat} nodeParams={vecLook1
vecPoint1 vecUp1}
      create node name={nodeTranslateReceptor} nodeType={translation}
nodeParams={translationReceptor} parent={nodeLook1}
            create node name={nodeRotReceptor} nodeType={rotationeuler}
nodeParams={rotationReceptor} parent={nodeTranslateReceptor}
                  create node name={nodeReceptor} nodeType={pdb}
nodeParams={receptor sphere stick nodeMode3hfl_fv}
parent={nodeRotReceptor}

      create node name={nodeTranslateLigand} nodeType={translation}
nodeParams={translationLigand} parent={nodeLook1}
            create node name={nodeRotLigand} nodeType={rotationeuler}
nodeParams={rotationLigand} parent={nodeTranslateLigand}
                        create node name={nodeLigand} nodeType={pdb}
nodeParams={ligand sphere stick nodeMode3hfl_ly} parent={nodeRotLigand}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VIEW-ova %%%%%
create view name={view1} nodeParams={nodeLook1}
```

Rezultat izvođenja skripte:



Slika 28. Prikaz PDB datoteka s bojanjem po lancima.

5.2. Višestruki pogledi

Test pokazuje način korištenja dva različita pogleda na istu scenu. Kao i u prethodnom testu, učitane su dvije PDB datoteke i razmaknute po *X*-osi, te obojane po lancima.

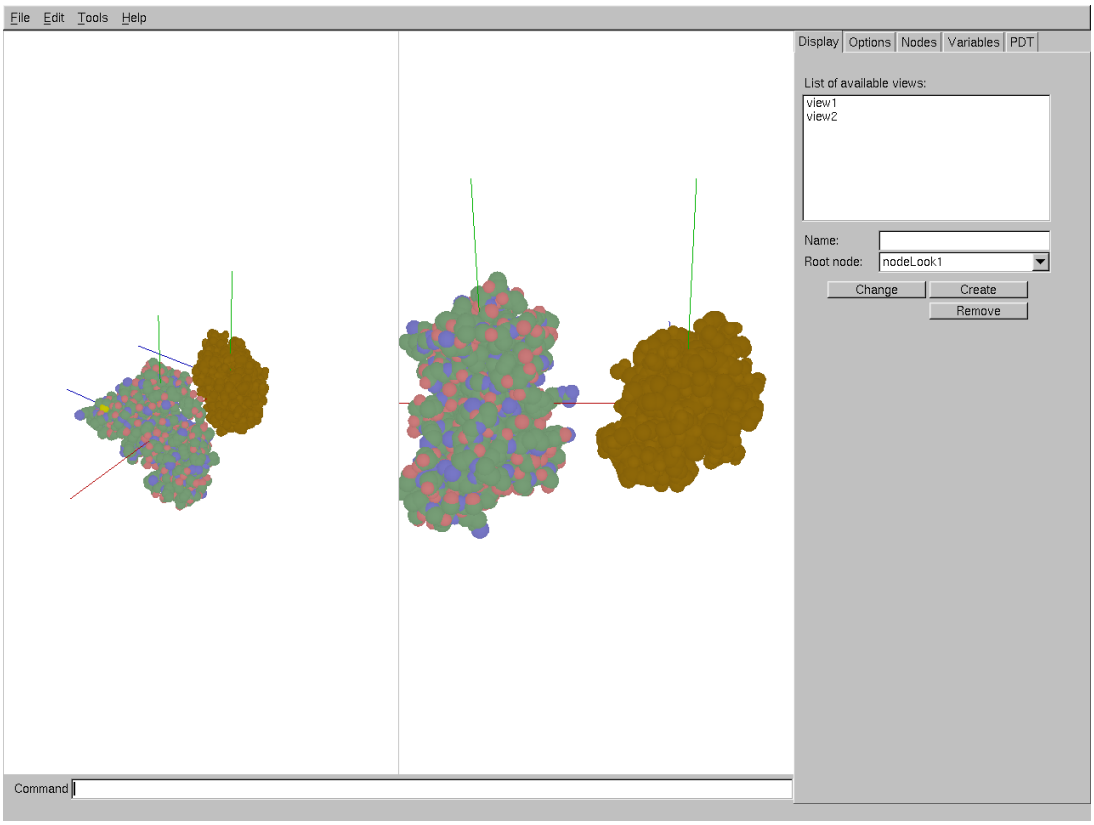
Skripta:

```
%%%%% UCITAVANJE PDB-ova %%%%%
create pdb path={data/pdb/3hfl_fv.pdb} name={receptor}
radiusPath={data/system/atmtypenumbers} radiusType={united} center={true}
centerType={mass}
create pdb path={data/pdb/3hfl_ly.pdb} name={ligand}
```

```
radiusPath={data/system/atmtypenumbers} radiusType={united} center={true}
centerType={mass}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% UCITAVANJE MESH-eva %%%%%
create mesh path={data/models/sphere.dat} name={sphere} type={dat}
center={true} autoScale={1.0}
create mesh path={data/models/stick.dat} name={stick} type={dat}
center={true} autoScale={1.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VARIJABLI %%%%%
create variable={nodeMode3hfl_fv} type={ulonglong}
value={0x0001000000011105}
create variable={nodeMode3hfl_ly} type={ulonglong}
value={0x0001000000014105}

create variable name={translationReceptor} type={vector3} value={25.0 0.0
0.0}
create variable name={rotationReceptor} type={eulervector3} value={0.0
0.0 0.0}
create variable name={translationLigand} type={vector3} value={-20.0 0.0
0.0}
create variable name={rotationLigand} type={eulervector3} value={0.0 0.0
0.0}

create variable name={vecLook1} type={vector3} value={240 198 -180.0}
create variable name={vecPoint1} type={vector3} value={0.0 0.0 0.0}
create variable name={vecUp1} type={vector3} value={0.0 1.0 0.0}

create variable name={vecLook2} type={vector3} value={0.0 90.0 -180.0}
create variable name={vecPoint2} type={vector3} value={0.0 0.0 0.0}
create variable name={vecUp2} type={vector3} value={0.0 1.0 0.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE NODE-ova %%%%%
create node name={nodeLook1} nodeType={lookat} nodeParams={vecLook1
vecPoint1 vecUp1}
```

```
      create node name={nodeTranslateReceptor} nodeType={translation}
nodeParams={translationReceptor} parent={nodeLook1}
            create node name={nodeRotReceptor} nodeType={rotationeuler}
nodeParams={rotationReceptor} parent={nodeTranslateReceptor}
                  create node name={nodeReceptor} nodeType={pdb}
nodeParams={receptor sphere stick nodeMode3hfl_fv}
parent={nodeRotReceptor}
      create node name={nodeTranslateLigand} nodeType={translation}
nodeParams={translationLigand} parent={nodeLook1}
            create node name={nodeRotLigand} nodeType={rotationeuler}
nodeParams={rotationLigand} parent={nodeTranslateLigand}
                  create node name={nodeLigand} nodeType={pdb}
nodeParams={ligand sphere stick nodeMode3hfl_ly} parent={nodeRotLigand}

create node name={nodeLook2} nodeType={lookat} nodeParams={vecLook2
vecPoint2 vecUp2}
      create node name={nodeTranslateReceptor1} nodeType={translation}
nodeParams={translationReceptor} parent={nodeLook2}
            create node name={nodeRotReceptor1} nodeType={rotationeuler}
nodeParams={rotationReceptor} parent={nodeTranslateReceptor1}
                  create node name={nodeReceptor1} nodeType={pdb}
nodeParams={receptor sphere stick nodeMode3hfl_fv}
parent={nodeRotReceptor1}
      create node name={nodeTranslateLigand1} nodeType={translation}
nodeParams={translationLigand} parent={nodeLook2}
            create node name={nodeRotLigand1} nodeType={rotationeuler}
nodeParams={rotationLigand} parent={nodeTranslateLigand1}
                  create node name={nodeLigand1} nodeType={pdb}
nodeParams={ligand sphere stick nodeMode3hfl_ly} parent={nodeRotLigand1}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VIEW-ova %%%%%
create view name={view1} nodeParams={nodeLook1}
create view name={view2} nodeParams={nodeLook2}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati izvođenja skripte:



Slika 29. Prikaz s dva različita pogleda na istu scenu.

5.3. Prikaz molekularne površine

Test pokazuje učitanu molekulu iz PDB datoteke kojoj je izračunata molekularna površina. Površina je u prikazu superponirana na prikaz molekule i to prozirno-plavom bojom, kako bi se mogao uočiti odnos molekule i njene površine. Molekula je prikazana modelom kugli i štapića.

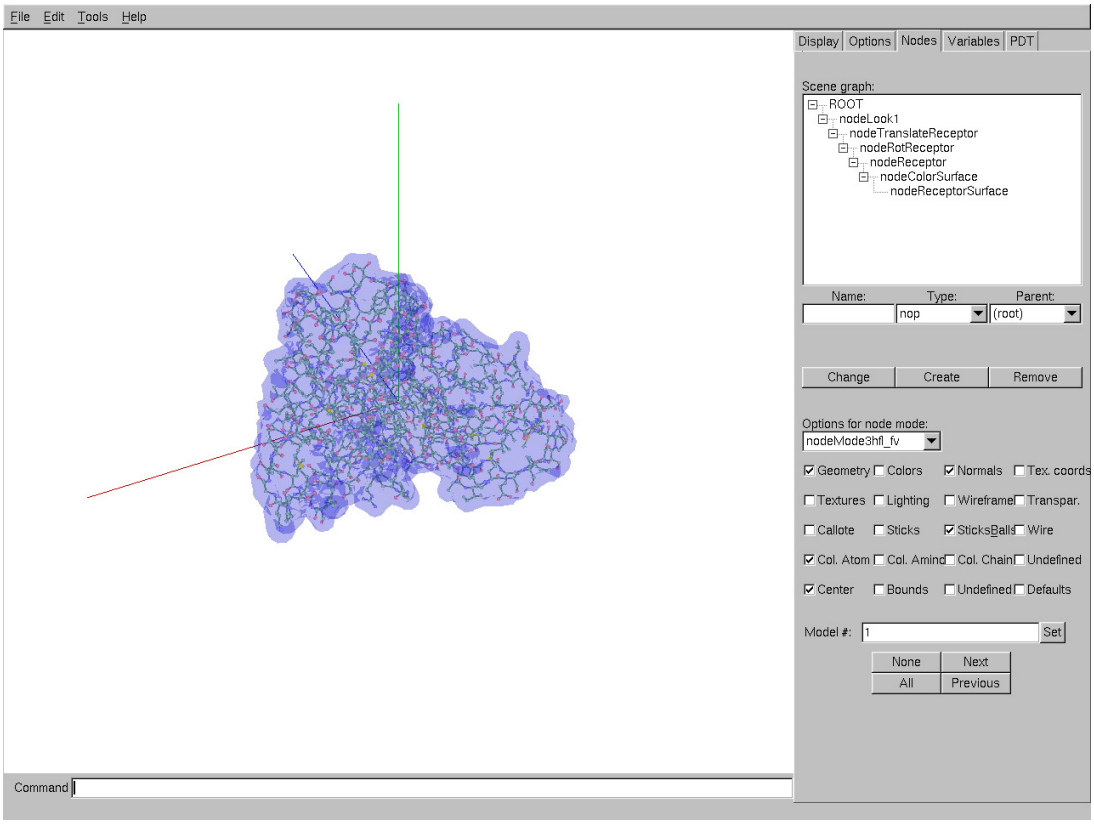
Skripta:

```
%%%%% UCITAVANJE PDB-ova %%%%%
create pdb path={data/pdb/3hfl_fv.pdb} name={receptor}
radiusPath={data/system/atmtypenumbers} radiusType={united} center={true}
centerType={mass}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%% UCITAVANJE MESH-eva %%%%%
create mesh path={data/models/sphere.dat} name={sphere} type={dat}
center={true} autoScale={1.0}
create mesh path={data/models/stick.dat} name={stick} type={dat}
center={true} autoScale={1.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VARIJABLI %%%%%
create variable={nodeMode3hfl_fv} type={ulonglong}
value={0x0001000000011405}
create variable={nodeMode3hfl_ly} type={ulonglong}
value={0x0001000000014105}
create variable={nodeModeMsReceptor} type={ulonglong}
value={0x0001000000011105}

create variable name={translationReceptor} type={vector3} value={0.0 0.0
0.0}
create variable name={rotationReceptor} type={eulervector3} value={0.0
0.0 0.0}
create variable name={colorSurface} type={color} value={0 0 255 75}

create variable name={vecLook1} type={vector3} value={47.0 96.0 -100.0}
create variable name={vecPoint1} type={vector3} value={0.0 0.0 0.0}
create variable name={vecUp1} type={vector3} value={0.0 1.0 0.0}

create ms name={receptor} path={data/ms/receptorMs}
value={nodeModeMsReceptor}
create mesh path={data/ms/receptorMs.atc} name={receptorSurface}
type={ms} center={false} autoScale{0.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE NODE-ova %%%%%
create node name={nodeLook1} nodeType={lookat} nodeParams={vecLook1
vecPoint1 vecUp1}
      create node name={nodeTranslateReceptor} nodeType={translation}
nodeParams={translationReceptor} parent={nodeLook1}
            create node name={nodeRotReceptor} nodeType={rotationeuler}
```

```
nodeParams={rotationReceptor} parent={nodeTranslateReceptor}
                  create node name={nodeReceptor} nodeType={pdb}
nodeParams={receptor sphere stick nodeMode3hfl_fv}
parent={nodeRotReceptor}
                        create node name={nodeColorSurface}
nodeType={color} nodeParams={colorSurface} parent={nodeReceptor}
                               create node name={nodeReceptorSurface}
nodeType={mesh} nodeParams={receptorSurface nodeModeMsReceptor}
parent={nodeColorSurface}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VIEW-ova %%%%%
create view name={view1} nodeParams={nodeLook1}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati izvođenja skripte:



Slika 30. Prikaz površine molekule superponirane početnom modelu.

5.4. Ocjenjivanje rezultata procesa prianjanja proteina

Test prikazuje tri molekule učitane iz PDB datoteka: receptor, ligand i točni kompleks. Receptor i ligand predstavljeni su crvenom i zelenom bojom, dok je kompleks obojan u prozirno-plavu. Razlog je odabira prozirne boje lakše uočavanje podudaranja konformacije liganda i receptora s točnim kompleksom. Osim izvršene skripte, za dobivanje rezultata prikazanog slikom bilo je potrebno učitati izlaznu datoteku procesa prianjanja i pokrenuti postupak ocjenjivanja rezultata.

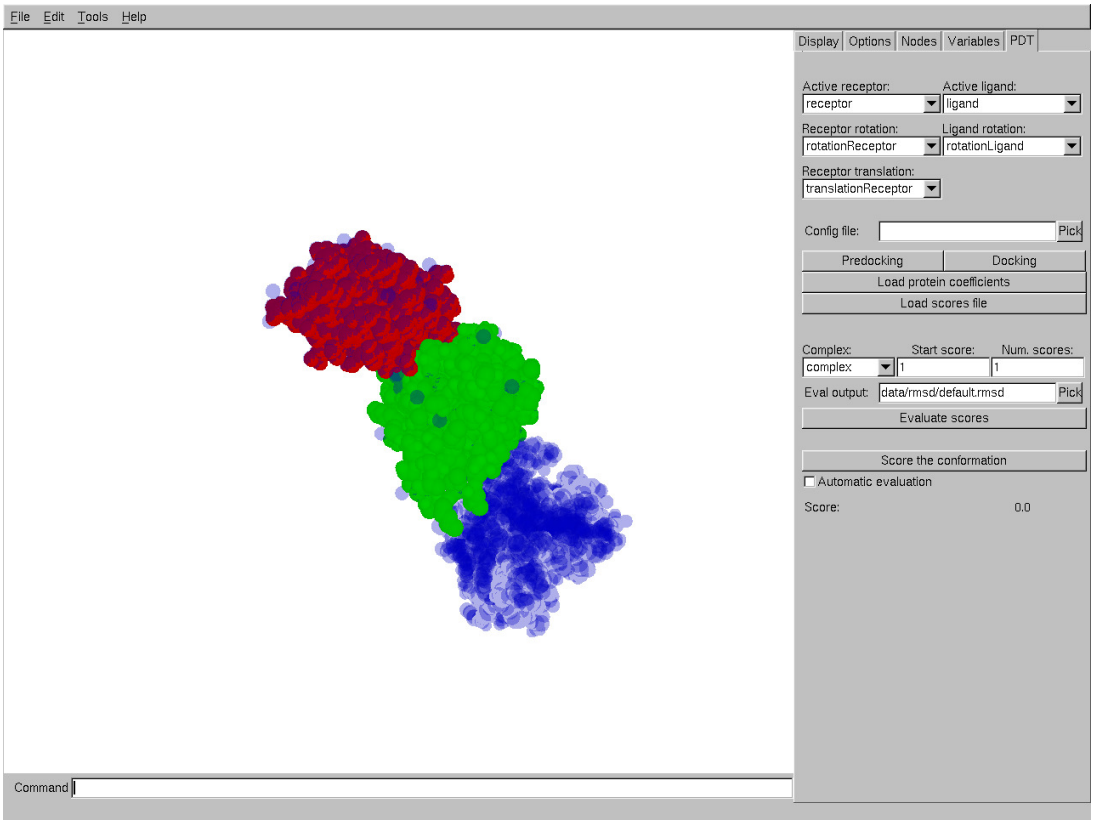
Skripta:

```
%%%%% UCITAVANJE PDB-ova %%%%%
create pdb path={data/pdb/3hfl_fv.pdb} name={receptor}
radiusPath={data/system/atmtypenumbers} radiusType={united}
center={false} centerType={mass}
create pdb path={data/pdb/3hfl_ly.pdb} name={ligand}
radiusPath={data/system/atmtypenumbers} radiusType={united}
center={false} centerType={mass}
create pdb path={data/pdb/3hfl.pdb} name={complex}
radiusPath={data/system/atmtypenumbers} radiusType={united} center={true}
centerType={mass}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% UCITAVANJE MESH-eva %%%%%
create mesh path={data/models/sphere.dat} name={sphere} type={dat}
center={true} autoScale={1.0}
create mesh path={data/models/stick.dat} name={stick} type={dat}
center={true} autoScale={1.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VARIJABLI %%%%%
create variable={nodeMode3hfl_fv} type={ulonglong}
value={0x0001000000000105}
create variable={nodeMode3hfl_ly} type={ulonglong}
value={0x0001000000000105}
create variable={nodeMode3hfl} type={ulonglong}
value={0x0001000000000105}
```

```
create variable={nodeModeCoordSystem} type={ulonglong} value={0x0005}

create variable name={translationReceptor} type={vector3} value={0.0 0.0
0.0}
create variable name={rotationReceptor} type={eulervector3} value={0.0
0.0 0.0}
create variable name={rotationLigand} type={eulervector3} value={0.0 0.0
0.0}

create variable name={colorLigand} type={color} value={255 0 0 255}
create variable name={colorReceptor} type={color} value={0 255 0 255}
create variable name={colorComplex} type={color} value={0 0 255 80}
create variable name={colorCoordSystem} type={color} value={255 255 0
255}

create variable name={vecLook1} type={vector3} value={-103.0 -47.0 -
180.0}
create variable name={vecPoint1} type={vector3} value={0.0 0.0 0.0}
create variable name={vecUp1} type={vector3} value={0.0 1.0 0.0}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE NODE-ova %%%%%
create node name={nodeLook1} nodeType={lookat} nodeParams={vecLook1
vecPoint1 vecUp1}
      create node name={nodeTranslateReceptor} nodeType={translation}
nodeParams={translationReceptor} parent={nodeLook1}
            create node name={nodeRotReceptor} nodeType={rotationeuler}
nodeParams={rotationReceptor} parent={nodeTranslateReceptor}
                  create node name={nodeColorReceptor} nodeType={color}
nodeParams={colorReceptor} parent={nodeRotReceptor}
                        create node name={nodeReceptor} nodeType={pdb}
nodeParams={receptor sphere stick nodeMode3hfl_fv}
parent={nodeColorReceptor}

      create node name={nodeRotLigand} nodeType={rotationeuler}
nodeParams={rotationLigand} parent={nodeLook1}
            create node name={nodeColorLigand} nodeType={color}
```

```
nodeParams={colorLigand} parent={nodeRotLigand}
                  create node name={nodeLigand} nodeType={pdb}
nodeParams={ligand sphere stick nodeMode3hfl_ly} parent={nodeColorLigand}

      create node name={nodeColorComplex} nodeType={color}
nodeParams={colorComplex} parent={nodeLook1}
            create node name={nodeComplex} nodeType={pdb}
nodeParams={complex sphere stick nodeMode3hfl} parent={nodeColorComplex}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% STVARANJE VIEW-ova %%%%%
create view name={view1} nodeParams={nodeLook1}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Rezultati izvođenja skripte:



Slika 31. Ocjenjivanje konformacije u odnosu na točan kompleks.

6. Upute za pokretanje

Alat je namijenjen izvođenju na *Linux* operativnom sustavu. Uspješno je testiran i pokrenut na računalu sa sljedećom konfiguracijom:

- procesor: Intel Core2Duo 9300, takta 2.5Ghz
- memorija: 2GB RAM
- grafička kartica: NVidia Quadro FX 570M
- operativni sustav: Ubuntu 9.04 32bit

Kako je Longview modul sustava Protein Docking Tool, on se kompajlira zajedno s ostalim dijelovima sustava. Prilikom razvoja ovog alata korišteno je razvojno okruženje *Eclipse Galileo*. Umjesto *Eclipse*-a, moguće je koristiti i zadnju verziju *gcc* kompajlera. Protein Docking Tool, pa tako i Longview ovisni su o nekoliko važnih biblioteka, koje se besplatno mogu preuzeti s Interneta. To su:

- GNU Scientific Library (GSL)
- GSL C Basic Linear Algebra Subprograms (GSL CBLAS)
- Boost Regex
- Open Graphics Library (OpenGL), v1.1
- OpenGL Utility Toolkit (GLUT)
- Fast Light Toolkit (FLTK), v1.3

7. Zaključak

Ovim radom opisan je alat izrađen u svrhu ostvarivanja ciljeva zadanih u uvodu. Alat, nazvan Longview, omogućuje učitavanje i prikaz PDB datoteka i površine molekula, isto kao i površina molekula rekonstruiranih iz koeficijenata sfernih harmonika mapiranih na grid. Učitane modele moguće je translatirati i rotirati, a nad učitanim PDB datotekama moguće je provoditi i proces ručnog prianjanja proteina. Također je moguće i pokretanje procesa automatskog prianjanja proteina. Ispravnost rada alata potvrđena je njegovom usporedbom s drugim alatima slične namjene (*Hex, Pymol, VMD*). No, osim tih ranije definiranih mogućnosti, ovaj alat nudi i mnogo više.

Alat je dizajniran prilično apstraktno, što omogućuje veliku prilagodljivost za raznolike primjene. Tako se, na primjer, isti alat može koristiti i za pregledavanje običnih trodimenzionalnih modela nevezanih uz molekule proteina, izvođenje jednostavnih grafičkih simulacija, i slično. Za razliku od ostalih alata slične namjene, osim interakcije u obliku translacije i rotacije korištenjem miša, moguće je mijenjati sve definirane varijable, što omogućuje promjenu boje, pogleda i raznih drugih postavki. Naredbe se također mogu zadavati i u tekstualnom obliku, što omogućuje izvršavanje kompleksnih skripti. Još jedna vrlo bitna razlika i prednost Longview-a u odnosu na ostale alate je mogućnost korištenja i do 4 prikaza u istom sučelju odjednom.

Iz svih navedenih mogućnosti vidljivo je kako je alat napravljen u sklopu ovog rada i više nego zadovoljio sve početne ciljeve, te ostvario i puno više od njih.

Longview je zamišljen kao modul već postojeće aplikacije, te je time i implementiran u sustav Protein Docking Tool. Njegova mogućnost primjene, osim u navedenom sustavu, je velika zbog njegove općenitosti i prilagodljivosti. Svoju primjenu može naći u farmaciji kao pripomoć pri nalaženju novih lijekova, kao preglednik molekula za kućnu upotrebu, ili za učitavanje i pregledavanje raznih formata trodimenzionalnih modela.

Postoje velike mogućnosti daljnjeg rada na ovom alatu. Zasigurno bi trebalo implementirati višedretvenost, što bi bilo vrlo korisno prilikom pokretanja procesorski i

vremenski zahtjevnih operacija kao što je prianjanje proteina, jer korisnika ne bi ometalo u njegovom radu s alatom. Pri tome, korisno bi bilo i u sučelju prikazivati trenutni postotak pretraženog prostora stanja, kao i trenutno stanje rang liste konformacija. Gledamo li na Longview sa strane preglednika modela, mogla bi se implementirati i mogućnost učitavanje više vrsta mesh modela.

Literatura

[1] Sović, I. *Vizualizacija makromolekula*. Završni rad. Fakultet elektrotehnike i računarstva. 2008.

[2] *Colors*. http://jmol.sourceforge.net/jscolors/ . Datum pristupa: 17.06.2010.

[3] Sayle, R. *Amino/Shapely colors*. http://life.nthu.edu.tw/~fmhsu/rasframe/SHAPELY.HTM . Datum pristupa: 17.06.2010.

[4] Sayle, R. *CPK Colors*. http://life.nthu.edu.tw/~fmhsu/rasframe/CPKCLRS.HTM . Datum pristupa: 17.06.2010.

[5] Khronos Group. *The industry standard for high performance graphics*. http://www.opengl.org/ . Datum pristupa: 17.06.2010.

[6] Foster G. *Understanding and implementing scene graphs*. http://www.gamedev.net/reference/programming/features/scenegraph/ . Datum pristupa: 17.06.2010.

[7] Spitzak, B. *Fast Light Toolkit (FLTK)* http://www.fltk.org/ . Datum pristupa: 17.06.2010.

[8] Ritchie, D. *Parametric protein shape recognition*. Doktorski rad. University of Aberdeen. 1998.

[9] Antulov-Fantulin, N., Čanadi, I., Piškorec, M., Sović, I. *PDT – Protein docking tool*. Projektna dokumentacija. 2009.

Sažetak

Alat za prianjanje proteina: modul za vizualizaciju

Ključne riječi: proteini, prianjanje, kugline funkcije, vizualizacija, molekula, Longview, graf scene

Postojeći sustav *Protein Docking Tool* (PDT) izvršava proces prianjanja proteina temeljen na razvoju površine u red kuglinih funkcija. Ulazne podatke, pojedine međukorake postupka kao i izlazne podatke korisno je vizualizirati zbog provjere točnosti i uvida u rad sustava. U sklopu ovog rada razvijen je alat nazvan *Longview* koji radi upravo tu funkciju. On omogućuje učitavanje PDB datoteka, trodimenzionalnih modela, konfiguriranje grafa scene za ostvarivanje željenog prikaza, te daje korisniku mogućnost upravljanja pomoću skriptnih naredbi. Longview također može rekonstruirati površinu molekule reprezentiranu koeficijentima kuglinih funkcija. Osim toga, alat nudi i mogućnost korištenja više pogleda istovremeno za prikaz konstruirane scene, te puno drugih opcija. Za razliku od ostalih alata slične namjene, Longview je specifičan upravo zbog svoje fleksibilnosti. Svaka od ovih mogućnosti detaljno je razmotrena u samom radu.

Alat je razvijan kao modul sustava PDT, te je uspješno implementiran u njega. Na samom kraju rada prikazano je testiranje alata, te iznesen zaključak i ideje za budući rad na njemu.

Summary

Protein docking tool: visualization module

Keywords: protein, docking, spherical harmonics, visualization, molecules, Longview, scene graph

Protein Docking Tool (PDT) is an existing system that performs the protein docking process based on the development of the protein surface into the order of spherical harmonics. Visualization of the input and output data, as well as the intermediate steps, gives us the insight to the way how this system works, and also gives us a chance to check the accuracy of the results. For this thesis, a tool called *Longview* was developed. This tool is able to load PDB files and threedimensional models, to configure the scene graf for the current display, and to give users the ability to control every aspect of the displayed scene using the script commands. Longview can also reconstruct the molecular surface represented through spherical harmonic coefficients. Also, this tool has the ability to create and use multiple views of the scene, as well as many other options. In comparison to other similar-purpose tools, Longview is specific because of its flexibility. All available options regarding this tool are discussed in more detail in the thesis itself.

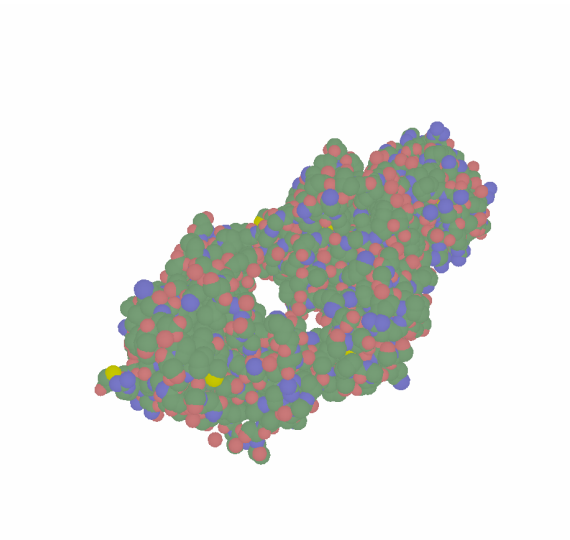
Longview was developed as a module of the PDT system, where it was successfully implemented. Tests performed on Longview are shown near the end of this thesis, followed by the conclusion and ideas for the future work on this project.

Privitak A - Popis boja korištenih u radu

U tablici 18 naveden je popis boja korištenih za označavanje pojedinih atoma u alatu razvijenom u sklopu ovog rada. Tablica 19 prikazuje popis korištenih boja za označavanje pojedinih aminokiselina, dok su boje korištene za pojedine lance izložene u tablici 20. Slike 32, 33 i 34 prikazuju molekule s primijenjenim bojanjima po atomima, aminokiselinama i lancima.

Tablica 18. Boje atoma korištene u Longview-u.

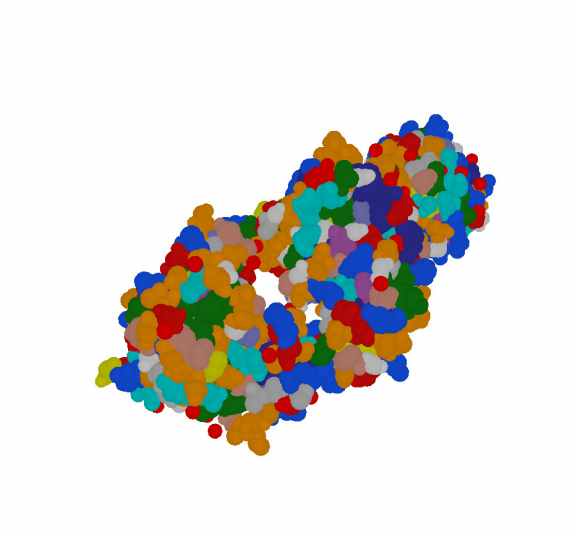
Oznaka atoma	**Crvena boja (*r*)**	**Zelena boja (*g*)**	**Plava boja (*b*)**	**Prozirnost (*a*)**
C	153	204	153	255
H	204	204	204	255
N	153	153	255	255
O	255	153	153	255
S	255	255	0	255
ostalo	230	230	230	255



Slika 32 . Prikaz molekule s bojanjem atoma po CPK sustavu.

Tablica 19 . Boje aminokiselina korištene u Longview-u.

Naziv aminokiseline	Crvena boja (*r*)	Zelena boja (*g*)	Plava boja (*b*)	Prozirnost (*a*)
ala	199	199	199	255
arg	20	89	255	255
asn	0	219	219	255
asp	230	10	10	255
cys	230	230	0	255
gln	0	219	219	255
glu	230	10	10	255
gly	235	235	235	255
his	130	130	209	255
ile	15	130	15	255
leu	15	130	15	255
lys	20	89	255	255
met	230	230	0	255
phe	51	51	168	255
pro	219	150	130	255
ser	250	150	0	255
thr	250	150	0	255
trp	179	89	179	255
tyr	51	51	168	255
val	15	130	15	255
asx	255	105	179	255
glx	255	105	179	255
ostalo	255	0	0	255

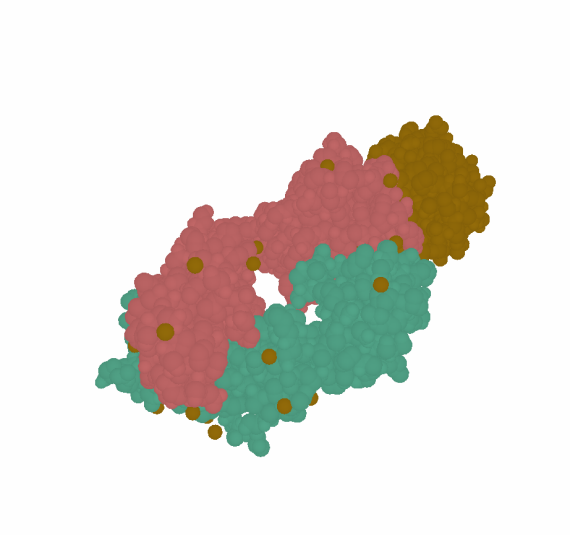


Slika 33 . Prikaz molekule s bojanjem po aminokiselinama.

Tablica 20. Boje lanaca korištene u Longview-u.

Oznaka lanca	**Crvena boja (*r*)**	**Zelena boja (*g*)**	**Plava boja (*b*)**	**Prozirnost (*a*)**
a	192	208	255	255
b	176	255	176	255
c	255	192	200	255
d	255	255	128	255
e	255	192	255	255
f	176	240	240	255
g	255	208	112	255
h	240	128	128	255
i	245	222	179	255
j	0	191	255	255
k	205	92	92	255
l	102	205	170	255
m	154	205	50	255
n	238	130	238	255
o	0	206	209	255
p	0	255	127	255

q	60	179	113	255
r	0	0	139	255
s	189	183	107	255
t	0	100	0	255
u	128	0	0	255
v	128	128	0	255
w	128	0	128	255
x	0	128	128	255
y	184	134	11	255
z	178	34	34	255
ostalo	255	255	255	255



Slika 34. Prikaz molekule s bojanjem po lancima.