SVEUČILIŠTE U ZAGBREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 155

Distribuirani sustav za vizualizaciju mikrobioloških podataka

Denis Tošić

Zagreb, siječanj 2011.

Hvala prof.dr.sc. Željki Mihajlović na pomoći i savjetima.

Hvala Dini, Zoranu, Viktoru, Vuletu i Čakiju
na svim ovim godinama koje smo proveli
skupa u školskim klupama, ali i izvan njih.

Hvala mami, tati i Franji što nikada nisu rekli 'Neću.'.

And thank you Raquel… for everything.

Sadržaj

1. UVOD

Vizualizacija podataka u današnje je vrijeme neophodna. Medicina, biologija, tehnološke znanosti – sve one iz dana u dan generiraju nevjerojatne količine informacija te je njihova analiza jednostavno nemoguća u njihovom osnovnom obliku. Najprirodniji i najjednostavniji način njihovog prikazivanja jest putem vizualizacijskih alata – bilo da se radi o grafovima, 3D modelima ili nečem drugom. Programska podrška koja se koristi u te svrhe mora biti izuzetno visoke kvalitete. Bilo da se radi o preciznosti izračuna, kvaliteti prikaza, brzini izvođenja ili mogućnosti kolaboracije; svaki od tih faktora važan je u današnjoj znanstvenoj zajednici.

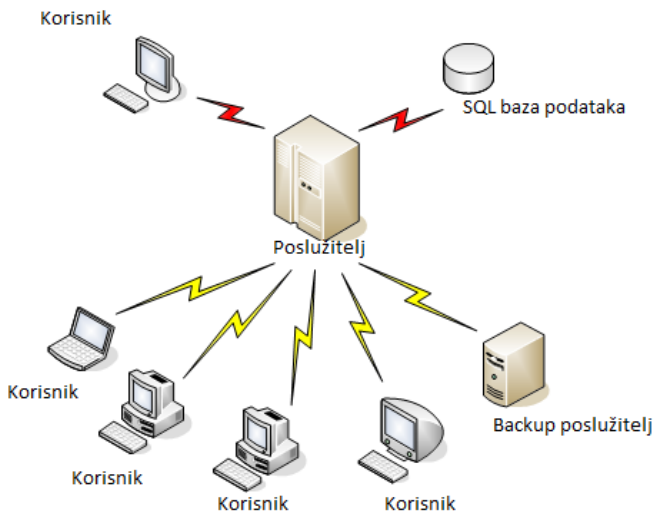
U slijedećim poglavljima prikazani su rezultati istraživanja i kreiranja takvog jednog sustava. Cilj je bilo stvoriti sustav za kolaboraciju, interakciju i pomoć oboljelima od cistične fibroze i njihovim liječnicima, ali i istraživačima. Temeljna svojstva koje bi sustav trebao posjedovati jesu distribuiranost, neovisnost o računalnoj platformi te mogućnost kvalitetne i brze vizualizacije medicinski i mikrobioloških podataka te interaktivnost s njima.

2. DISTRIBUIRANI, VIŠEPLATFORMSKI GRAFIČKI SUSTAVI

Glavni zadatak u okviru ovog diplomskog rada bila je izrada vizualizacijskog sustava za prikaz bioloških i mikrobioloških podataka. No uz ovaj problem, dodatni uvjet bila je i neovisnost konačne programske podrške o operacijskom sustavu te mogućnost pristupa putem Internet preglednika. Stoga je posebna cjelina posvećena upravo ovoj tematici.

Distribuirani sustav, u svojem najširem smislu riječi, jest sustav koji se sastoji od više autonomnih procesa koji međusobno komuniciraju u svrhu realizacije zajedničkog cilja. U kontekstu ovoga rada definicija se može ograničiti i zamijeniti slijedećom: distribuirani sustav jest sustav kod kojeg se mrežom prenose podatci od klijentskog računala prema poslužitelju. Poslužitelj obrađuje dobivene podatke, te rezultat(e) obrade prosljeđuje dalje ili ih vraća klijentu.

Najjednostavniji model distribuiranog sustava jest sustav dvaju računala (ili dvaju procesa na istom računalu). Ovisno o konkretnom problemu, jedno računalo može imati ulogu klijenta, a drugo poslužitelja ili oba računala mogu biti ravnopravna. Potonjim se želi reći da oba računala mogu poprimiti uloga klijenta ili poslužitelja, ovisno o trenutnoj potrebi. Vrlo blizak pojam distribuiranom računarstvu jest paralelno računarstvo, koje uključuje i paralelne programe. Paralelno programiranje nije bilo uključeno u procesu izrade ovog rada.

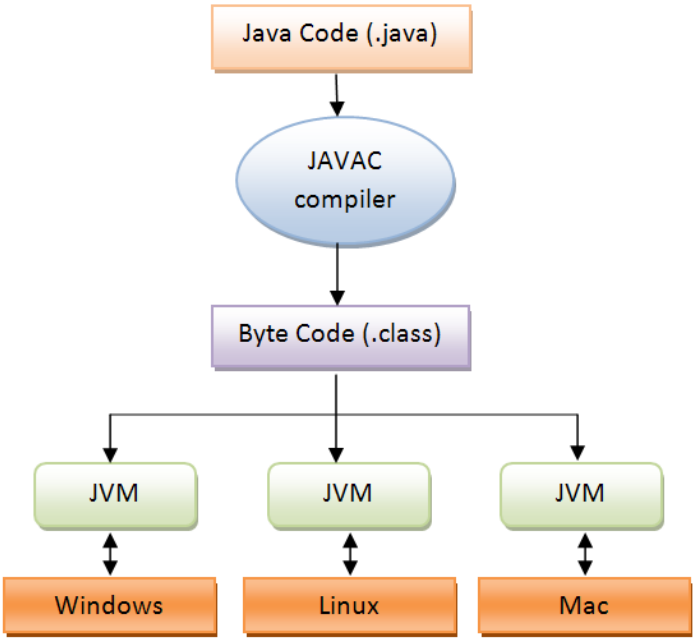


Slika 1. Primjer distribuiranog računalnog sustava.

Višeplatformski sustavi s druge strane jesi računalni sustavi koji obavljaju svoj zadatak neovisno o računalnoj platformi na kojoj se izvršavaju. Računalne platforme razlikuju se po svojoj arhitekturi (npr. 32-bitna naspram 64-bitne), operacijskom sustavu (Microsoft Windows, Linux, Mac OS X, Solaris itd.), programskim jezicima koje podržavaju (C#, Java, JavaScript itd.) te pripadajućim korisničkim sučeljima (npr. grafička sučelja naspram tekstualnih). Najvažnije obilježje višeplatformskog sustava jest mogućnost njegovog izvršavanja unutar različitih operacijskih sustava. Takvo svojstvo definira se u fazi dizajniranja sustava, prije nego se krene u samu implementaciju. Postoji više načina putem kojih se navedeno svojstvo može postići. Dva najčešća i ujedno najjednostavnija jesu programiranje u programskom jeziku Java te izrada aplikacije koja će se odvijati unutar web preglednika.

Glavna značajka programskog jezika Java u kontekst višeplatformskih sustava jest da se izvršni java programi izvršavaju unutar JVM-a, virtualnog stroja koji izvršava sav kod napisan u Java jeziku. Dotični se program može izvršavati unutar različitih operacijskih sustava, pod uvjetom da imaju instalirani JVM. Drugim riječima, Java izvršne datoteke koji sadrže Java bytecode se ne izvršavaju izravno unutar operacijskog sustava, već na višoj razini. Iako se na ovaj način postiže visoka razina prenosivosti koda, ova filozofija ima i svoje nedostatke - Java ne podržava izravne sistemske pozive (API operacijskog sustava na kojem se izvršava). Kako bi se ovaj problem riješio, Java raspolaže JNI (JavaNativeInterface) sučeljem kojim se omogućava izravan pristup sistemski funkcijama. Nažalost, korištenjem JNI se ponovno narušava prenosivost koda, jer npr. neki sistemski pozivi koji rade unutar Windows operacijskog sustava ne postoje unutar Linux operacijskog sustava.

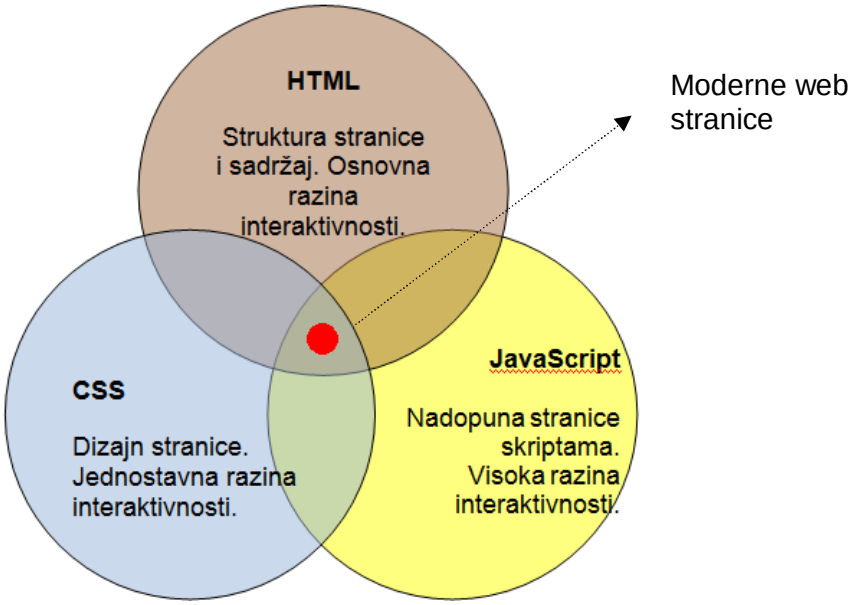
Drugi način korištenja Java programskog jezika u svrha izrade višeplatformskog sustava jesu Java appleti. Java appleti izvršavaju se unutar Internet preglednika korištenjem JVM-a. Uvedeni su još u prvoj verziji Java 1995. godine te su obično pisani u Javi, no mogu biti i unutar drugih jezika koje se kompajliraju u Java bytecode (Jython, JRuby itd.). Appleti se koriste u svrhu pružanja interaktivnih funkcija web sadržajima koje standardni HTML ne podržava (primjeri 3D prikaza, interakcija mišem, prikazivanje videa i sl.).



Slika 2. Model rada JVM-a.

Appleti se po svojoj brzini izvršavanja mogu mjeriti sa C++-om, iako su uglavnom sporiji, a daleke 1995. bili su osjetno brži od JavaScripta i pružali mnogo veću funkcionalnost. Međutim, od tada su se HTML i JavaScript uvelike razvili, te podržavaju gotovo sve ono što podržavaju appleti (koji su i dalje ograničeni sistemskim pozivima), a također se i po pitanju brzine izvršavanja previše ne razlikuju - Java je i dalje brža, no ne toliko osjetno, a za svoje izvršavanje korisnici i dalje trebaju imati instaliran JRE što u nekim (primjerice lokalni korporacijskim sustavim) nije dozvoljeno zbog sigurnosnih razloga.

Treći način ostvarenja višeplatformskog sustava jest korištenje Internet preglednika te HTML-a i JavaScripta, bez ikakvih dodataka (Slika 3.). Na taj se način korisniku omogućava pristup sustavu s bilo koje platforme koja ima Internet preglednik koji podržava JavaScript. No ne samo to - sustav također može biti distribuiran na bilo koji način, jer se njime pristupa putem Internet preglednika, i sve što je potrebno jest odgovarajuće sučelje za komunikaciju s poslužiteljima.



Slika 3. Najsažetiji opis web stranice koja podržava grafiku i interaktivnost visoke razine.

Postaje očito da za problem kreiranja distribuiranog, višeplatformskog sustava postoje dva rješenja: korištenje Java appleta ili HTML-a i JavaScripta. S obzirom da je naglasak konačnog sustava na grafičkoj komponenti, odgovor bi prije nekoliko godina bio prilično jednostavan: Java appleti. Naime, do nedavno HTML nije pružao učinkovito, interaktivno sučelje za realističnu, trodimenzionalnu grafiku, s iznimkom flash-a. Bilo je pokušaja popularizirati tehnologije poput VRML-a, X3D ili SVG-a, no niti jedna od njih uspjela jače zaživjeti.

Međutim, World Wide Web konzorcij (W3C) u suradnji sa *Web Hypertext Application Working Group* (WHATWG) razvija novi HTML standard, tzv. HTML5 koji donosi nove mogućnosti te bogatiju sintaksu ponajviše na multimedijskom području. Primjerice, uvedene su nove vrste elemenata (<video>, <audio>, <canvas>) kojima se olakšava integracija multimedijskih sadržaja unutar web stranica, a također se veliku pažnju posvetilo i unapređenju semantičkog sadržaja (<section>, <article>, <header>).

Odgovor na pitanje koje je rješenje bolje nije lako dati, niti je ono jednoznačno. Da bi se do njega došla valja detaljno razmotriti sve faktore koji će utjecati na konačni sustav, ali i sam sustav: koja je njegova primjena, tko će se njime koristiti i u kakvim uvjetima, koliko mora biti siguran, u kolikoj mjeri distribuiran i slično. U nastavku će biti razmotren upravo takav jedan realni sustav, te će se postepeno analizirati njegovi zahtjevi i ograničenja, a

kraju će biti dano i konačno obrazloženje zašto je odabrana jedna od navedenih tehnologija.

3. BIOINFORMATIKA I BIOINFORMATIČKI SUSTAVI

Bioinformatika jest primijenjena znanost koja povezuje statistiku i računarsku znanost s molekularnom biologijom, počela se naglo razvijati 80-tih godina prošlog stoljeća, zbog potrebe sekvenciranja DNK (sekvenciranje jest postupak određivanja redoslijeda nukleotidnih baza – adenina, guanina, citozina i timina).

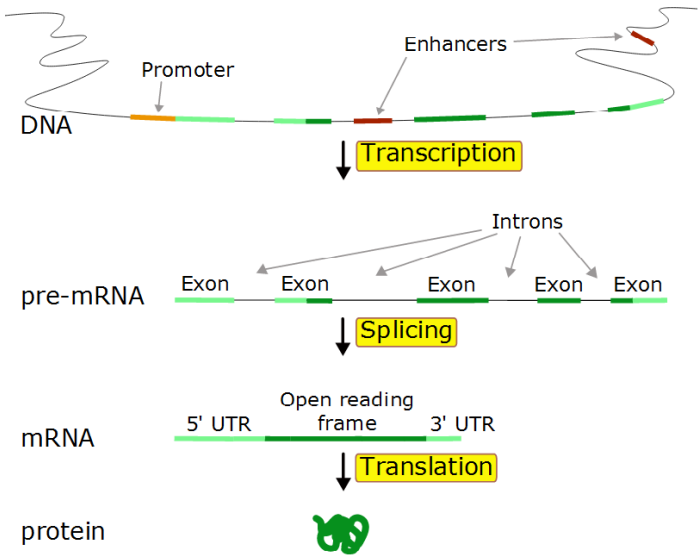
Glavni zadatak bioinformatike jest proširenje znanja o biološkim procesima. Stoga se bioinformatičari bave razvojem i unapređivanjem baza podataka, algoritama, računarskih i statističkih tehnika te teorije kako bi se riješili formalni i praktični problemi koji izranjanju analizom bioloških podataka. Neki od glavnih problema kojima se bavi bioinformatika jesu analiza sekvenci DNA i proteina, njihova usporedba s drugim sekvencama i proteinima te njihova (trodimenzionalna) vizualizacija. Nadalje tu još spada i pronalaženje gena, dizajniranje lijekova, pronalaženje lijekova, protein-protein međudjelovanja, modeliranje evolucije itd.

Ono što bioinformatiku razlikuje od drugih istraživačkih bioloških disciplina jest usredotočenost na razvoj i primjenu računarski zahtjevnih metoda (raspoznavanje uzoraka, dubinske pretrage podataka, strojno učenje i vizualizacija) pri ostvarivanju svojih ciljeva. Ovakav pristup u potpunoj je suprotnosti s eksperimentalnim, laboratorijskim istraživanje.

Bioinformatika bila je prvi puta primijenjena prilikom stvaranja i održavanja bioloških baza podataka kada su se počele istraživati sekvence nukleotida i amino kiselina. Razvoj takvih baza nije podrazumijevao samo njihovo dizajniranje, već i neprestan razvoj sučelje i metoda unosa i ažuriranja podataka. Primjerice, kako bi se mogle istraživati stanične reakcije prilikom različitih faza nekog oboljenja ili infekcije, biološki podatci moraju biti opisani i povezani na način da se uvidom u njih može pojmiti jasna slika onoga što se dogodilo i što se trenutno događa u stanici.

U tom pogledu bioinformatika se razvila u područje koje danas izučava interakciju i međuovisnost ne samo sekvenci gena i aminokiselina, već i strukture proteina i kako ona

utječe na stanične procese. Sam proces analiziranja i interpretiranja podataka naziva se računalna biologija. Osim toga u bioinformatiku još spadaju i problemi poput razvoja i implementacije alata za učinkovito rukovanje raznim oblicima informacija te razvoj novih i unapređivanje postojećih algoritama, statistički postupci čiji je cilj pronaći položaj gena u sekvenci, predvidjeti strukturu/funkciju proteina i sl. Laički rečeno, bioinformatika želi računalnim putem što bolje objasniti prikaz na Slika 4 te sve posljedice koje proizlaze iz nje. Promoter označava početak sekvence koju teba prevesti, dok su pojačivači (eng. *enhancers*) elementi koju povečavaju učestalost transkripcije (sam proces još uvijek nije do kraja razjašnjen).



Slika 4. Generalizirani prikaz stvaranja proteina transkripcijom.

3.1. ANALIZA SEKVENCI (SEQUENCE ANALYSIS)

Analiza sekvenci jest sustavno analiziranje sekvenci gena u svrhu daljeg istraživanja. Ovaj postupak podrazumijeva različite akcije nad danom sekvencom DNA ili proteina:

- Slaganje i uspoređivanje sekvenci (eng. *squence alignment search*),
- Pretraživanje i nadopunjavanje sekvenci unutar baze podataka (eng. *sequence databases search*),
- Pronalaženje sekvenci koje se ponavljaju (eng. *repeated sequence search*).

Tijekom devedesetih godina prošlog stoljeća došlo je naglog poraste brzine otkrivanja novih sekvenci gena. Međutim samo otkrivanje sekvenci kao funkcionalnih skupina samo po sebi ne pruža detaljan uvid u njihovu ulogu u biološkom organizmu. Međutim, uspoređivanje postojećih sekvenci s poznatom funkcionalnošću s novim, nepoznatim sekvencama jest jedan način istraživanja biološkog organizma na molekularnoj razini. Upravo to i jest uloga analize sekvenci – definirati funkcionalnost gena i proteina analiziranje i uspoređivanjem sekvenci.

Analiza sekvenci u sklopu bioinformatike i molekularne biologije jest automatizirani proces i izvodi se na računalu. Uključuje nekoliko različitih postupaka:

- Uspoređivanje sekvenci kako bi se pronašle sličnosti i razlike između njih.
- Identificiranje te distribucija introna, eksona i regulatornih elemenata.
  - introni su dijelovi DNA koji se ne prevode u proteine postupkom translacije
  - eksoni su dijelovi DNA koji se prevode u proteine postupkom translacije
  - regulatorni elementi su dijelovi DNA koji definiraju način na koji se informacije sadržane u genim prevode u konačne produkte (proteine)
- Pronalaženje i uspoređivanje jednostrukih mutacija (eng. *single point mutation*).
  - Jednostruke mutacije su vrsta mutacije kada dolazi do promjene jednog nukleotida unutar sekvence. Takve mutacije omogućuju raznolikost među jedinkama iste vrste, i važne su u pogledu otpornosti na različite bolesti te reakcije na lijekove i cjepiva.

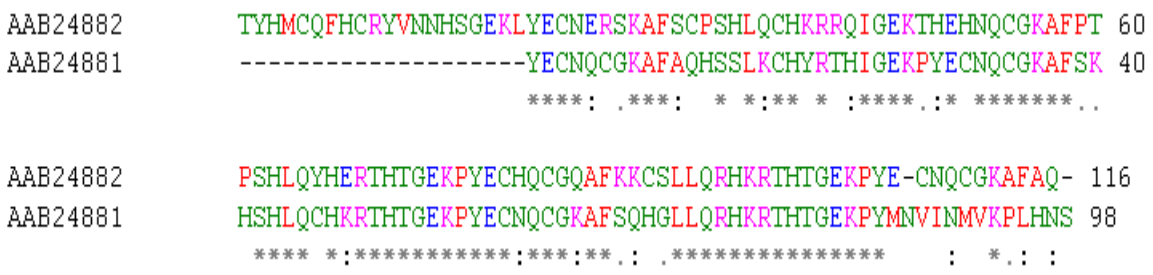
3.2. SLAGANJE SEKVENCI

Slaganje sekvenci jest postupak organiziranja DNA, RNA ili proteina u sekvence kako bi se identificirala područja sličnosti koja mogu biti posljedice funkcionalne, strukturalne ili evolucijske povezanosti između promatranih sekvenci. Takvo slaganje (zapravo uspoređivanje) ima slijedeću interpretaciju: ako dvije promatrane sekvence imaju istog predaka (bilo da govorimo o organizmu, stanici, DNA ili proteinu), tada razlike između njih možemo interpretirati kao jednostruke mutacije, a praznine kao mutacije brisanja ili dodavanja koje su nastale u jednoj ili obje promatrane sekvence od trenutka kada je došlo do njihovog razdvajanja (dioba vrste ili stanice, multipliciranje DNA). U slučaju da je

riječ o analizi proteina, jednakost aminokiselina na određenom mjestu nam grubo govori o tome koliki je stupanj očuvanosti (rasprostranjenosti) neke sekvence unutar cijelog potomstva. Ako je ta sekvence očuvana u velikoj mjeri među potomstvom, bilo da je riječ o biološkoj vrsti, ili o molekulama unutar jednog organizma, onda je ona motiv (sequence motif).

Rezultat slaganja sekvenci predstavlja se grafičko – tekstualnom kombinacijom. Pojedine sekvence se zapisuju u retke, na način da analogni parovi pripadaju istom stupcu. Na **slici** prikazana je tekstualna reprezentacija slaganja. Svako slovo predstavlja točno određenu aminokiselinu, svaka boja opisuje dodatne specifikacije (bazičnost, kiselost, veličinu, hidrofobnost itd.). Na lijevoj strani nalazi se ime sekvence, a na desnoj duljina. Ispod sekvence naneseni su razni markeri od kojih svaki ima svoje značenje:

- „ - “ – praznina, ne postoji aminokiselina na tom mjestu
- „ ***** “ – identičan zapis
- „ **:** “ – supstitucija kiselinom iz iste skupine
- „ **.** “ – supstitucija kiselinom sličnog prostornog oblika



Slika 5. Primjer slaganja dvije sekvence.

Na sličan definirano je i obilježavanje sekvenci nukleotida (gena), s time da postoje samo četiri baze: adenin, citozin, guanin i timin.

3.2.1. Metode slaganja

Iako na prvi pogled slaganja sekvenci izgleda izrazito jednostavno, praktične probleme nije moguće riješiti u razumnom roku samo ljudskom rukom. Zbog toga se razvijaju i primjenjuju složeni algoritmi koji automatski provode slaganje. Računalne metode slaganja

sekvenci dijele se u dvije glavne skupine: globalno slaganje i lokalno slaganje. Globalno slaganje jest jedan oblik globalne optimizacije koje „grubom silom“ traži rješenje. Drugim riječima, on promatra cijele sekvence i na temelju njih izrađuje usporedbu. Lokalno slaganje s druge strane pokušava pronaći segmente sekvence koji su slični ili identični, iako cijele sekvence globalno mogu biti vrlo različite. Iako je lokalno slaganje često bolji izbor, ono može biti sporije zbog faze pronalaženja sličnih regija sekvenci. I globalne i lokalne metode ubrzavaju se i poboljšavaju na različite načine: dinamičkim programiranjem te heurističkim ili vjerojatnosnim algoritmima.

Globalno slaganje daje dobre rezultate u slučaju kada postoji jedan minimalan stupanj sličnosti te kada su sekvence koje se uspoređuju približno jednake duljine. Jedan od poznatijih globalnih algoritama jest Needleman-Wunsch algoritam koji se temelji na dinamičkom programiranju. Algoritam uspoređuje dvije sekvence, A i B. Postupak se temelji na pronalaženju najbolji parova iz zadanih sekvenci.

Na početku je zadana matrica sličnosti *S* koja može biti slijedećeg oblika:

Nukleotidna Base	**Adenin**	**Guanin**	**Citozin**	**Timin**
Adenin	10	-1	-3	-4
Guanin	-1	7	-5	-3
Citozin	-3	-5	9	0
Timin	-4	-3	0	8

Iz nje se da očitati da parovi (*A*,*A*), (G, G), (C, C) i (T, T) imaju pozitivne vrijednosti, a ostali vrijednosti manje ili jednake nuli, što znači da je slaganje po principu „isto s istim“ povećava kvalitetu konačnog slaganja sekvenci. Također je očito da se preferiraju parovi (A, A) nad npr. (C, C) . Slijedeći korak jest izgradnja matrice *F* dimenzija *mxn* gdje su *m* i *n* duljine sekvenci, a element *F*(*i, j*) sadrži numeričku vrijednost dobrote slaganja elemenata A(i) i B(j). Algoritam najprije popunjava matricu *F*, a zatim iz nje i danih sekvenci izračunava optimalno slaganje. U nastavku je dan pseudokod algoritma.

```
//izračun matrice F
//d - penalty gap
for i=0 to length(A)
    F(i,0) ← d*i
  for j=0 to length(B)
      F(0,j) ← d*j
for i=1 to length(A)
    for j = 1 to length(B)
    {
      Match ← F(i-1,j-1) + S(Ai, Bj)
      Delete ← F(i-1, j) + d
      Insert ← F(i, j-1) + d
      F(i,j) ← max(Match, Delete, Insert)
}

//izračun konačnog slaganja
  AlignmentA ← ""
  AlignmentB ← ""
  i ← length(A)
  j ← length(B)
  while (i > 0 and j > 0)
  {
    Score ← F(i,j)
    ScoreDiag ← F(i - 1, j - 1)
    ScoreUp ← F(i, j - 1)
    ScoreLeft ← F(i - 1, j)
    if (Score == ScoreDiag + S(Ai, Bj))
    {
      AlignmentA ← Ai + AlignmentA
      AlignmentB ← Bj + AlignmentB
      i ← i - 1
      j ← j - 1
    }
    else if (Score == ScoreLeft + d)
    {
      AlignmentA ← Ai + AlignmentA
      AlignmentB ← "-" + AlignmentB
      i ← i - 1
    }
    otherwise (Score == ScoreUp + d)
    {
      AlignmentA ← "-" + AlignmentA
      AlignmentB ← Bj + AlignmentB
      j ← j - 1
    }
  }
```

```
while (i > 0)
{
  AlignmentA ← Ai + AlignmentA
  AlignmentB ← "-" + AlignmentB
  i ← i - 1
}
while (j > 0)
{
  AlignmentA ← "-" + AlignmentA
  AlignmentB ← Bj + AlignmentB
  j ← j - 1
}
```

Lokalno slaganje se primjenjuje kod sekvenci koja su globalno različite, ali unutar sebe sadrže veće regije sličnost. Za takvo uparivanje koristi se Smith-Waterman algoritam. Ovaj algoritam je također jedan od općenitijih pristupa rješavanju problema lokalnog slaganja i također se temelji na dinamičkom programiranju. Glavna ideja ovog algoritma jest da on promatra segmente svih mogućih duljina, ne samo cijelu sekvencu, te na temelju pojedinih usporedbi odabire najbolje slaganje. Već je u ovom trenutku očito kako je ovako pristup izuzetno sporiji od globalnog. Ovaj algoritam koristi matricu zamjene, naspram Needleman-Wunsch algoritama koji koristi matricu sličnosti. Matrica zamjene odražava vjerojatnost zamjene jedne baze/aminokiseline drugom (prijelaz iz jednog stanja u drugo). Matrica zamjene *S* prema Smith-Waterman algoritmu izgrađuje se na slijedeći način:

S(i, 0) = 0, (0 ≤ i ≤ m)
S(i, 0) = 0, (0 ≤ i ≤ m)

IF a_i = b_j THEN w(a_i, b_j) = w(pogodak)
OR IF a_i != b_j THEN w(a_i, b_j) = w(promašaj)

$$S(i,\ j) = \max \begin{Bmatrix} 0 \\ S(i-1,\ j-1) + w(,) & Match/Mismatch \\ S(i-1, j) + w(,-) & Deletion \\ S(i, j-1) + w(-,) & Insertion \end{Bmatrix},$$

,

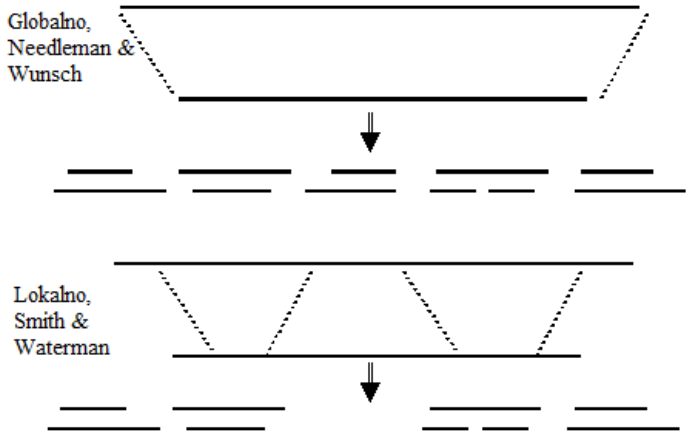
$$1 \le i \le m,\ 1 \le j \le n$$

gdje je:

a i b nizovi skupa ∑ (alfabet)
m = duljina(a)
n = duljina(b)
w (c, d) – težinska funkcija

Ono što je zanimljivo (i bitno) kod ove matrice jest da se konačne vrijednosti elemenata nalaze u rasponu [0, ∞ >. Ovo je upravo razlog zašto ovaj algoritam pogoduje lokalnom slaganju – nema kaznenih (negativnih) vrijednosti što lokalnim sličnosti omogućuje da dođu do izražaja. Nakon što se konstruira matrica sličnosti *S* između dvije date sekvence *a* i *b,* primjenjuje se prilagođeni backtrack algoritam kako bi se pronašla optimalno slaganje.

Na slici prikazana je temeljna razlika lokalnog i globalnog pristupa nad manje sličnim sekvencama - kvaliteta lokalnog algoritma se očituje u manjem broju prekida u konačnom slaganju.



Slika 6. Globalno naspram lokalnog slaganja.

Osim ova dva temeljna algoritma, postoje još mnogi drugi (noviji, brži i sofisticiraniji) koji ne spadaju u okvir ovog rada te stoga neće biti obrađeni. Međutim ono što jest važno u kontekstu ove tematike jesu matrice zamjene koje ti algoritmi koriste te su one najvažnije nakratko opisane u slijedećem poglavlju.

3.2.2. Matrice zamjene

U okviru bioinformatike i evolucijske biologije matrice zamjene predstavljaju empirijski zapis promjenjivosti gena/proteina. U svojem najosnovnijem obliku, one predstavljaju matricu vjerojatnosti prijelaza iz jednog stanja u drugo, slično matricama prijelaza u okviru Markovljevih lanaca. Elementi matrice S popunjavaju se prema slijedećoj formuli:

$$S_{i,j} = \log\frac{p_i * M_{i,j}}{p_i * p_j} = \log\frac{M_{i,j}}{p_j} = \log\frac{mjerena\ frekvencija\ mutacije}{očekivana\ frenkvencija\ mutacije}, \quad (3.1.)$$

gdje su *pi* i *pj* očekivane vjerojatnosti pojavljivanja baze/aminokiseline *i* i *j*, dok je *Mi,j* vjerojatnost mutacije baze/kiseline *i* u *j*. Baza logaritma može poprimiti proizvoljne vrijednosti, te se često događa da je ista matrica zamjene zapisana u različitim bazama. U praksi se najčešće odabire ona koja omogućuje jednostavniji zapis elemenata, npr. takva baza da su svi elementi cjelobrojni.

Dvije najpoznatije vrste matrica zamjene su PAM (eng. Point Accepted Mutation) matrice te BLOSUM (eng. Block Substitution Matrix) matrice. PAM1 matrica izgrađena je na temelju uspoređivanja i promatranja vrlo ”bliskih” proteina koji se razlikuju u samo 1% svojih elemenata, a te su vrijednosti dobivene upravo navedenom formulom. Kako bi se uspoređivale jedinke koje nisu u bliskom srodstvu (razlike veće od 1%), matrica PAM1 se potencira do određenog stupnja *n* te se tada koristi matrica PAMn kao mjerilo usporedbe sličnosti. Najprimjenjivanije matrice su PAM30 i PAM70.

PAM matrice nisu se pokazale točnima u slaganju sekvenci koje su evolucijski gledano jako udaljenje (u početku su se koristile čak PAM250 matrice koji su davale potpuno netočne rezultate). Razlike između takvih sekvenci je teško aproksimirati velikim brojem malih promjena (potenciranjem osnovne PAM1 matrice). Zbog tog nedostatka uvedene su BLOSUM matrice.

BLOSUM matrice su konstruirane na temelju višestrukih slaganja sekvenci evolucijski udaljenih proteina. Vrijednosti matrice određenu su na temelju blokova sličnih sekvenci unutar višestrukog slaganja sekvenci. Naime, sekvence koji su imale postotak sličnosti *n* ili više (potpuno jednake regije), grupirane su i označene kao jedna skupina. Nakon toga se matrica sličnosti izgrađivala uspoređivanjem parova sekvenci tako da nisu razmatrani parovi visokog stupnja sličnosti te time umanjujući utjecaj sličnih sekvenci. Indeks *n* matrice BLOSUMn označavan upravo taj granični postotak sličnosti. Stoga BLOSUM matrice s višim indeksima služe za uspoređivanje bliskih jedinki, dok one sa nižim indeksom za uspoređivanje evolucijski udaljenih jedinki. Za izračun elemenata matrice zamjene *S* (nakon provedenog grupiranja) koristi se slijedeća formula:

$$S_{i,j} = \left(\frac{1}{\lambda}\right)\log\left(\frac{p_{i,j}}{q_i * q_j}\right), \tag{3.2.}$$

Ovdje *pij* označava vjerojatnost zamjene aminokiseline *i* sa aminokiselinom *j*, a *qi* i *qj* jesu vjerojatnosti pronalaženja aminokiselina *i* i *j* na bilo kojem mjestu u nekoj sekvenci. Faktor

λ samo služi kao faktor skaliranja, kako bi elementi matrice *S* sadržavali što „jednostavnije elemente“, tj. cjelobrojne vrijednosti. Najčešće korištena matrica jest BLOSUM62 koji ima slijedeći oblik (prikazana je samo donja dijagonalna matrica):

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Slika 7. BLOSUM62 matrica

Glavne razlike između PAM i BLOSUM matrica jesu slijedeće:

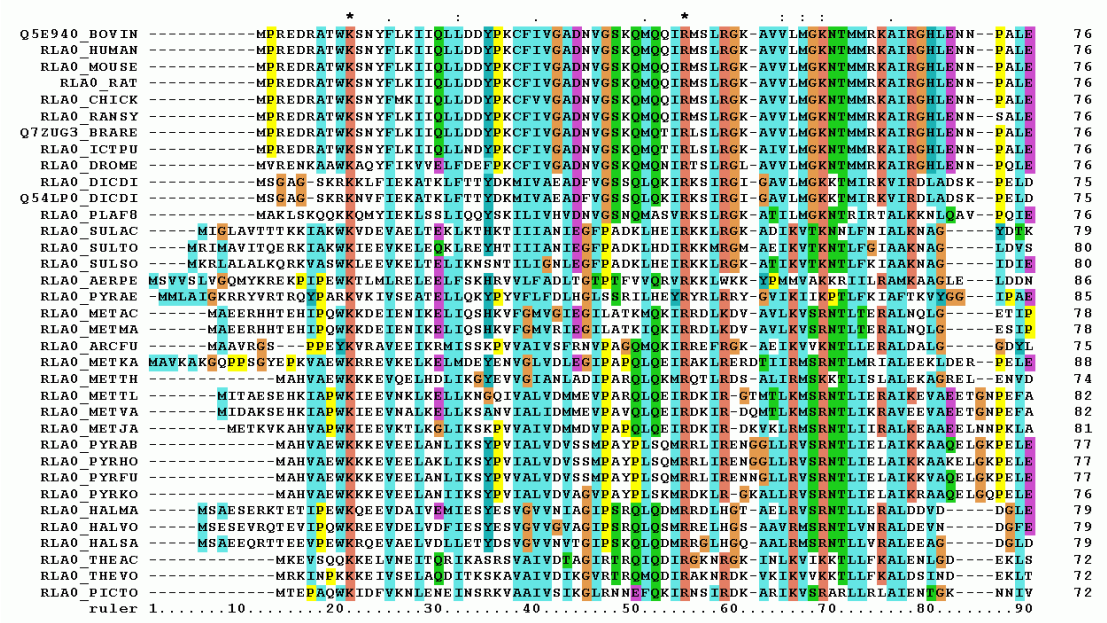
- PAM matrice izgrađene su na temelju eksplicitnih vrijednosti, dok su BLOSUM matrice temeljene na implicitnom modelu (grupiranje sličnih sekvenci).
- PAM matrice temelje se na mutacijama koji se očituju u globalnom modelu, što uključuje i regije koje su izuzetno podložne mutaciji i one kod kojih je vjerojatnost mutacije izuzetno mala. BLOSUM matrice se naspram toga temelje na regijama sekvenci visokog stupnja očuvanosti koje ne sadržavaju „rupe“.
- Viši indeksi unutar PAM matrice ukazuju na veći stupanj različitosti, dok je kod BLOSUM matrica obrnuto.

3.2.3. Višestruko slaganje

Osim izbora globalnog, odnosno lokalnog algoritma u okviru slaganja sekvenci, korisnik se također mora odlučiti o veličini skupa kojeg će analizirati i obrađivati. Naime, osim slaganje dvije sekvence, moguće je također i slagati više sekvenci odjednom (što je također i spomenuto u prethodnom poglavlju). Takvo se slaganje tada naziva višestruko slaganje.

Višestruko slaganja najčešće se koristi prilikom identificiranja očuvanih regija sekvenci (motiva) nad skupom sekvenci za koje se pretpostavlja da evolucijski srodne. Pronalazak takvih regija može biti indikator da je regija (ili neki njezin dio) katalitički aktivan dio enzima (što naravno nije točno), a također mogu pomoći u izgradnji evolucijskog stabla. Višestruko slaganje računalno je vrlo zahtjevno te se većina formalnog opisa svodi na NP-kompletne kombinatorne probleme. Unatoč tomu, višestruko slaganje pokazalo se vrlo korisnim i učinkovitim u slučaju jednoznamenkastog broja sekvenci.

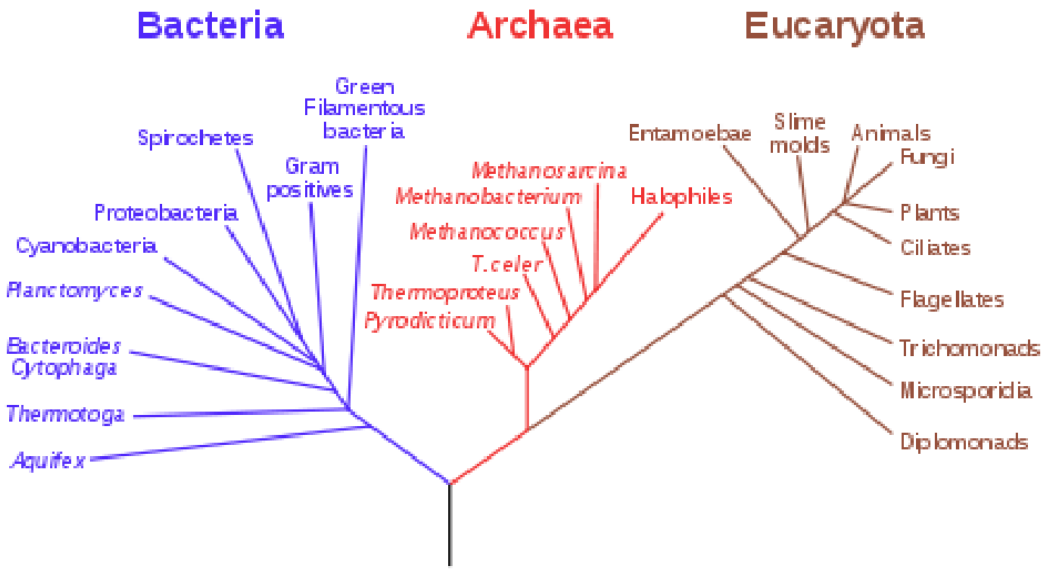
U većini se slučajeva za sekvence koje se obrađuju ovom metodom smatra ili pretpostavlja da su evolucijski povezane, tj. da imaju zajedničkog pretka. Na Slika 8 prikazan je primjer višestrukog slaganja ribosomnog proteina P0 (L10E) pronađenog u različitim organizmima. Razlike u sekvencama rezultat su jednostrukih mutacija, bilo da se radi o promjeni aminokiseline ili njenog brisanja ili dodavanja. Ovaj postupak se upravo koristi u svrhu pronalaženja motiva , odnosno evolucijski očuvanih regija.



Slika 8. Primjer višestrukog slaganja

Višestruko se slaganja, zbog svoje računalne zahtjevnosti, najčešće temelji na heurističkim metodama, jer globalno pretraživanje presporo konvergira prema rješenju (ili jednom od mogućih rješenja). Upravo zbog tog razloga kao temeljni matematički model ovog postupka koriste se skriveni Markovljevi lanci.

Neizravna korist višestrukog sekvencijskog slaganja jest mogućnost konstruiranja evolucijskog stabla. Postoje dva razloga tomu: prvi jest taj što višestruko slaganja omogućuje pronalaženje motiva, a drugi što takvim postupkom možemo iskoristiti postojeće znanje o nekim sekvencama ili njenim regijama. Naime, ako nam je poznata funkcionalnost određene regije u nekoj sekvenci, ako sličnu ili identičnu funkcionalnost pronađemo u nekoj drugoj sekvenci te ako utvrdimo da slaganje sekvenci daje kvalitetan rezultat, možemo naslutiti da su promatrane sekvence u određenom pogledu srodne. Analiziranjem velikog broja sekvenci možemo razviti filogenetsko stablo čiji primjer je prikazan na Slika 9.



Slika 9. Filogenetsko stablo života na Zemlji

3.3. DRUGA VAŽNA PODRUČJA

Osim analize i slaganja sekvenci, postoje i druga važna područje bioinformatike koja su ukratko opisana u nastavku ovog poglavlja.

3.3.1. Pronalaženje gena

Pronalaženje gena jest jedna od disciplina bioinformatike koja se bavi algoritamskim identificiranjem sekvenci koje su biološki funkcionalne i aktivne. To se posebice odnosi na gene zaslužne za stvaranje proteina, ali i na mnoge druge elemente poput RNA i regulatornih regija. Pronalaženje gena jest jedna od prvih i najvažnijih zadaća nakon što je

provedeno sekvenciranje. Određivanje da li je neka sekvenca funkcionalna nije isto što i određivanje funkcionalnosti gena. Naime, potonje iziskuje dodatna ispitivanja poput „brisanja“ danog gena iz sekvence kako bi se utvrdila promjena funkcionalnosti sekvence.

Postoje dvije metodologije pronalaženja gena: ekstrizična (temeljena na dokazima) te *ab intio* (dolaženje do zaključaka korištenjem samo računalnih metoda bez ekstrizične usporedbe s postojećim podatcima). U ekstrizičnom pristupu, unutar promatranog genoma se traže sekvence koje bi proizvele pronađene mRNA (eng. messanger RNA) ili proteine. Za danu mRNA, trivijalno je odrediti koja ju je DNA sekvenca proizvela, dok se za dani protein može odrediti familija sekvenci koja bi dovela do njegove sinteze. Jednom kada se definira takva sekvenca/e, dani se genom pretražuje za dobivenom sekvencom.

Ovakav pristup je poprilično jasan sam po sebi, no međutim, nije učinkovit koliko se to čini. Da bi se ovaj pristup sistematski primijenio, potrebno je sekvencirati mRNA i proteine, što postaje sve skuplje i skuplje proporcionalno s naprednošću danog organizma. Osim što samo sekvenciranje iziskuje mnogo vremena, i samo pretraživanje također traje. Naime, treba uzeti u obzir da je neke gene moguće pronaći samo u određenim tipovima stanica, ili u samo određenom trenutku razvoja organizma. Primjeri, neki ljudski geni izraženi su samo dok je čovjek u fazi embrija ili fetusa, čiji istraživanje za sobom povlači i moralne nedoumice.

Ekstrizični pristup je zbog svoje cijene i trajanja jednostavno nametnuo da se pronađe učinkovitiji pristup koji će biti automatiziran. Ta potreba rezultirala je *ab initio* pristupom koji sistematski po DNA sekvencama traži znakove (regulacijske regije) koji bi upućivali da je neki gen zaslužan za sintetiziranje proteina. *Ab initio* koristi heurističke metode i pretpostavke, te on zapravo samo naslućuje da je neki gen funkcionalan, što se utvrđuje naknadnim testiranjem.

Rezultati ove metode se uvelike razlikuju ovisno o tome da li su domena istraživanja prokarioti (organizmi bez stanične jezgre) ili eukarioti (organizmi sa staničnom jezgrom). U slučaju prokariota geni posjeduju specifične i relativno dobro poznate regulatorne regije, koje je lako uočiti automatiziranim putem.

Kada je riječ o eukariotima, kvaliteta rezultata opada ovisno o složenosti organizma. Regulatorne regije su manje istražene (a i puno složenije) od onih u prokariota te ih je stoga i teže otkriti. Mehanizam transkripcije (Slika 4.), zbog kojeg dolazi do kreiranja

eksona i introna, također otežava ovaj postupak jer duljina eksona naspram introna može biti relativno mala (nekoliko stotina naspram nekoliko tisuća nukleotida), što uvelike usporava pretraživanje za funkcionalnim elementima.

3.3.2. Komparativna genomika

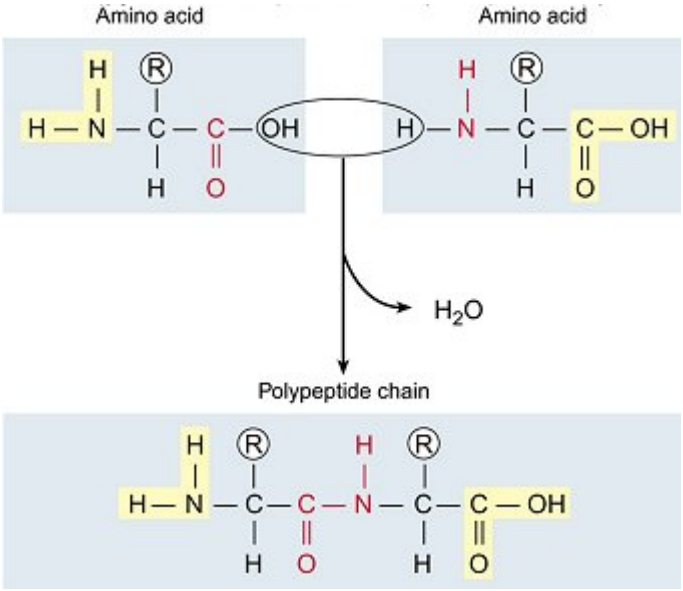
Komparativna genomika jest znanost o odnosima strukture i funkcionalnosti genoma (najčešće DNA) između biološki različitih vrsta ili rodova. Njezin glavni cilj jest produbiti znanje o biološkoj evoluciji (kao i vezama među vrstama) temeljem analize i usporedbe genoma različitih vrsta. Ogromnost i količina podataka zahtijevaju da metode komparativne genomike budu što više automatizirane. Pronalaženje gena (osnovnih jedinica nasljeđivanja) jest jedno od glavnih područja primjene komparativne genomike.

Komparativna genomika istražuje i sličnosti i razlike među proteinima, RNA te regulatornim područjima različitih organizama kako bi se došlo do saznanja kako proces selekcije (dominiranje jačeg nad slabijem) utječe na te elemente. Oni elementi koji su uzrok sličnosti između različitih vrsta bi se trebali očuvati kroz vrijeme (stabilna selekcija), dok bi oni koji uzrokuju razlike trebali divergirati u velikoj mjeri. Konačno, postoje i elementi koji su nebitni za evolucijski uspjeh organizma te za njih smatra da je selekcija neutralna.

Jedan od glavnih zadataka koje komparativna genomika sebi postavlja jest otkrivanje mehanizma nastanka eukariota (organizmi čije stanice imaju jezgru). Takav zadatak jest međutim teško rješiv s obzirom na veliki stupanj divergiranosti i veliki broj mutacija koje su se dogodile od pojave eukariota. Zbog toga komparativna genomika temelji svoje istraživanje na jednostavnijim organizmima kako bi otkrila osnovne mehanizme evolucije.

3.3.3. Predviđanje strukture proteina

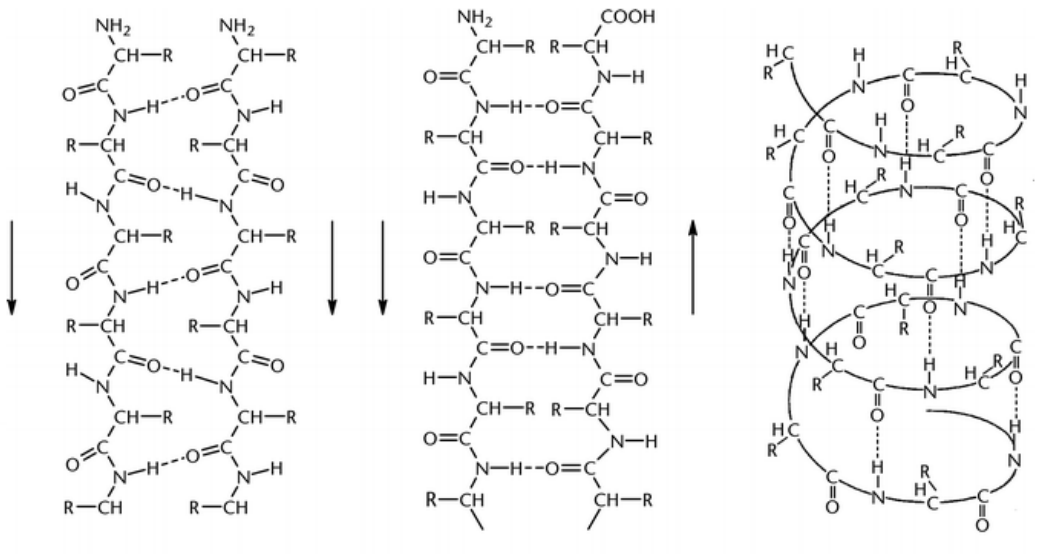
Predviđanje strukture proteina jest predviđanje njegove trodimenzionalne strukture iz zadanog lanca aminokiselina, tj. iz njegove primarne strukture. Aminokiseline su vezane u polipeptidni lanac peptitdnim vezama (-CONH-). Peptidna veza jest kovalentna veza nastala vezanjem karboksilne skupine (-COOH) te aminoskupine ($-NH_2$) uz nastajanje jedne molekule vode. Proces nastajanja peptidne veze prikazan je na Slika 10.



Slika 10. Općeniti proces nastajanja proteina; R označava preostali dio aminokiseline

Sekvenca aminokiselina predstavlja primarnu strukturu proteina koja u svojoj najjednostavnijoj formi čini lanac. Međutim, primarna struktura se vrlo rijetko pojavljuje kao jednostavni lanac (uglavnom se glavni lanac nadovezuju druge skupine i manji lanci)te ona stoga nije dovoljna za detaljan opis trodimenzionalne građe proteina. Zbog toga se za opis građe proteina koriste još sekundarna, tercijarna te kvartarna struktura.

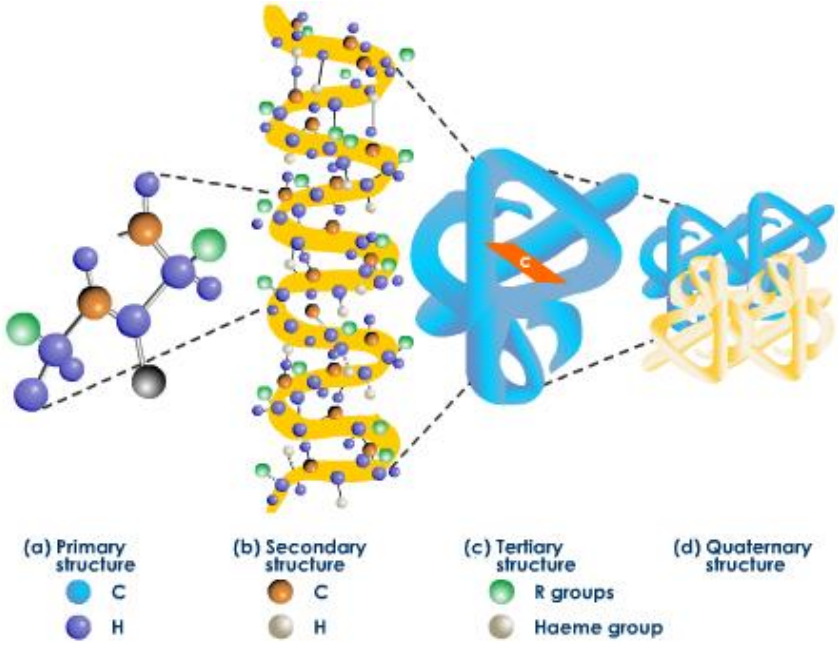
Sekundarna struktura jest općeniti trodimenzionalni prikaz proteina ili nukleinskih kiselina (DNA i RNA). On međutim ne odgovara na pitanje koje su točne pozicije atoma unutar molekule (odgovor na to daje tercijarna struktura), već definira samo vrijednosti za neke kutove unutar molekule (one vezane za glavni lanac) te vodikove veze između pojedinih elemenata molekule. U slučaju proteina, najčešće sekundarne strukture su alfa heliks zavojnice te beta plohe. Beta plohe su složene strukture izgrađene povezivanjem beta nizova vodikovim vezama. Ovisno o orijentaciji nizova, one mogu paralelne ili neparalelne. Na Slika 11Slika 11 dani su primjeri sekundarnih struktura.



Slika 11. Osnove sekundarne strukture – paralelna i neparalelna beta ploha te heliks zavojnica.

Tercijarna struktura jest trodimenzionalna struktura koju protein poprima nakon postupka pregibanja, te ona opisuje točne atomske koordinate. Pregibanje je fizikalni proces promjene oblika molekule pod utjecajem njezinih vlastitih, međuatomskih sila te drugih proteina. Ovaj proces odvija se spontano za vrijeme ili nakon sinteze molekule s ciljem postizanja minimuma Gibbsove slobodne energije. Naspram tercijarne, kvartarna struktura opisuje raspodjelu i strukturu više pregibanih proteina koji su vezani nekovalentnim vezama. Odnos primarne, sekundarne, tercijarne te kvartarne strukture prikazan je na Slika 12.

Previđanje strukture proteina je od fundamentalne važnosti u medicini (dizajniranje lijekova) te biotehnologiji (dizajniranje bioloških katalizatora – enzima). Predviđanjem sekundarne strukture pokušava se utvrditi koje regije proteina sadrže zavojnice, koje plohe itd. Točnost previđanja utvrđuje se uspoređivanjem s~~a~~ rezultatima koje daje DSSP (*eng. Define Secondary Structure of Proteins*) algoritam koji se primjenjuje na kristalnu strukturu proteina. Najbolje metode predviđanja sekundarne strukture dosežu točnost i do 80%.



Slika 12. Odnos primarne, sekundarne, tercijarne te kvartarne strukture

Predviđanje tercijarne strukture proteina svodi se na pronalaženje minimuma Gibbsove slobodne energije na temelju same sekvence aminokiselina. Ovaj zadatak jest računalno vrlo zahtjevan zbog činjenice da je broj mogućih prostornih struktura neke sekvence gotovo beskonačan. Ovom problemu može se doskočiti uspoređivanjem s~~a~~ drugim sličnim (homolognim) proteinima, za koje je poznata konačna tercijarna struktura, te za koje se pretpostavlja da imaju sličnu strukturu kao promatrani protein. Time se prostor pretraživanja drastično smanjuje (no i dalje ostaje velik).

U slučaju predviđanja kvartarne strukture proteina problem se svodi na predviđanje interakcije dvaju ili više proteina te njihove konačne strukture. Predviđanje ove strukture izuzetno je važno za razumijevanje međustanične komunikacije te raznih biokemijskih procesa.

4. STRUKTURE I MIKROBIOLOŠKIH PODATAKA I MOLEKULARNA GRAFIKA

Bioinformatika obiluje zaista velikom količinom podataka. Brojnost bioloških vrsta, gena, proteina i kemijskih reakcija jednostavno postavlja zahtjev za racionalnim i učinkovitim oblikovanjem struktura i baza u kojima će biti spremljene te informacije. Nadalje, te informacije potrebno je i prikladno prikazati, sa više ili manje detalja, i fokusom na sadržaj koji je bitan. Ovo poglavlje posvećeno je upravo ovoj tematici.

4.1. STRUKTURE I BAZE PODATAKA

U svojoj suštini, većina struktura podataka (vrste datoteka) su tekstualnog tipa. U najvažnije spadaju:

- FASTA i RAW – zapisivanje sekvenci
- MSF – višestruko slaganje sekvenci
- PDB – strukture proteina
- SBML, MML – formati temeljeni na XML-u
- i mnogi drugi…

Raznovrsnost vrsta zapisa leži u tome da ne postoji standardizacijski dokument o ovom problemu. Još tijekom prošlog desetljeća svatko tko je dizajnirao format koji najviše odgovara njegovom programu i njegovoj percepciji problema. Tako da filogenetičari preferiraju svoj format, korisnici određenih programskih paketa svoj itd. Međutim, svaki od tih formata je relativno rasprostranjen te su stoga još uvijek u upotrebi.

FASTA i RAW jesu tekstualni formati koji koriste se za zapis nukleotidnih ili peptidnih sekvenci. Gradivni elementi (nukleotidi ili aminokiseline) označavaju se veliki tiskani slovima latinske abecede. Jedina razlika između ova dva formata jest zaglavlje – ono u RAW formatu ne postoji. Zaglavlje se koristi za dodatni opis sekvence - ime, biološka vrsta i drugi detalji. Redovi zaglavlja započinju „>“ ili „;“ znakom, što označava da se radi o komentaru. Praksa je da prvi redak zaglavlja uvijek započinje sa „>“, a ostali koji slijede sa „;“.

```
>SEQUENCE_1
;human
MTEITAAMVKELRESTGAGMMDCKNALSETNGDFDKAVQLLREKGLGKAAKKADRLAAEG
LVSVKVSDDFTIAAMRPSYLSYEDLDMTFVENEYKALVAELEKENEERRRLKDPNKPEHK
IPQFASRKQLSDAILKEAEEKIKEELKAQGKPEKIWDNIIPGKMNSFIADNSQLDSKLTL
MGQFYVMDDKKTVEQVIAEKEKEFGGKIKIVEFICFEVGEGLEKKTEDFAAEVAAQL
>SEQUENCE_2
;rabit
SATVSEINSETDFVAKNDQFIALTKDTTAHIQSNSLQSVEELHSSTINGVKFEEYLKSQI
ATIGENLVVRRFATLKAGANGVVNGYIHTNGRVGVVIAAACDSAEVASKSRDLLRQICMH
```

Interpretacija korištenih kratica prikazana je u Tabela 1.

Tabela 1. Interpretacija kodova u FASTA formatu.

Kod nukleinske kiseline	Interpretacija	Kod aminokiseline	Interpretacija
A	adenozin	A	alanin
C	citozin	B	aspagarinska kiselina ili aspagarin
G	guanin	C	cistein
T	timin	D	aspagarinska kiselina
U	uracil	E	glutaminska kiselina
R	G ili A (purinske baze)	F	fenilalanin
Y	T ili C (pirimidinke baze)	G	glicin
K	G ili T (ketoni)	H	histidin
M	A ili C (amino skupine)	I	izoleucin
S	G ili C (jaka interakcija)	K	lizin
W	A ili T (slaba interakcija)	L	leucin
B	G ili T ili C (B nakon A)	M	metionin
D	G ili A ili T (D nakon C)	N	aspagarin
H	A ili C ili T (H nakon G)	O	pirolizin
V	G ili C ili A (V nakon T, U)	P	prolin
N	A ili G ili C ili T (bilo koji)	Q	glutamin
X	maskiran	R	arginin

-	rupa	S	serin
		T	treonin
		U	selenocistein
		V	valin
		W	triptofan
		Y	tirozin
		Z	glutaminska kiselina ili glutamin
		X	bilo koji
		*	prekid translacije
		-	rupa

FASTA format jako je raširen među bioinformatičarima, podržan je od strane većine programa za analizu sekvenci i jednostavan je za manipulaciju, no nepraktičan je za čitanje višestrukog slaganja (posebice ljudskim okom) s obzirom sekvence nisu poravnate te ne eksplicitno ne podržava dodavanje drugih informacija o sekvenci (osim putem komentara).

Upravo zbog ovog nedostatka, najvažniji (i najkorišteniji) format za višestruko slaganje sekvenci jest MSF (eng. *Multiple Sequence aligment Format*). Njegova prednost jest što je jednostavan za modifikaciju i čitanje (posebice ljudima) s obzirom da su sekvence poravnate, može sadržavati dodatne podatke (primjerice težine) te je barem djelomično podržan od većine programa. Primjer ispod prikazuje razlike u FASTA i MSF formatu prilikom zapisa višestrukog slaganja sekvenci. Razlika u „kvaliteti“ zapisa jest više nego očita.

```
//MSF

 Name: 3sil             Len:   276  Check: 6095  Weight:  1.00
 Name: 1kit             Len:   276  Check: 4858  Weight:  1.00
//

3sil  ---------- ---------- ---------- ---------- ----------
1kit  ALFDYNATGD TEFDSPAKQG WMQDNTNNGS GVLTNADGMP AWLVQGIGGR

3sil  ---------- ---------- ---------- ---------- ----------
1kit  AQWTYSLSTN QHAQASSFGW RMTTEMKVLS GGMITNYYAN GTQRVLPIIS

3sil  ---------- ---------- ---------- ---------- -KSVVFKAEG
1kit  LDSSGNLVVE FEGQTGRTVL ATGTAATEYH KFELVFLPGS NPSASFYFDG

3sil  EHFTD----- --QKGNTIVG SGSGGTT--- ---------- --KYFRIP--
1kit  KLIRDNIQPT ASKQNMIVWG NGSSNTDGVA AYRDIKFEIQ GDVIFRGPDR

3sil  ----AMCTTS KGTIVVFADA RHNTASDQSF IDT---AAAR STDGGKTWNK
1kit  IPSIVASSVT PGVVTAFAEK RVGGGDPGAL SNTNDIITRT SRDGGITWDT

3sil  KIAIYNDRVN SKLSRVMDPT CIVANI
1kit  ELNLTEQINV SDEFDFSDPR PIYD--
```

```
//FASTA

>3sil
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ---------- -KSVVFKAEG
EHFTD----- --QKGNTIVG SGSGGTT--- ---------- --KYFRIP—-
----AMCTTS KGTIVVFADA RHNTASDQSF IDT---AAAR STDGGKTWNK
KIAIYNDRVN SKLSRVMDPT CIVANI
>1kit
ALFDYNATGD TEFDSPAKQG WMQDNTNNGS GVLTNADGMP AWLVQGIGGR
AQWTYSLSTN QHAQASSFGW RMTTEMKVLS GGMITNYYAN GTQRVLPIIS
LDSSGNLVVE FEGQTGRTVL ATGTAATEYH KFELVFLPGS NPSASFYFDG
KLIRDNIQPT ASKQNMIVWG NGSSNTDGVA AYRDIKFEIQ GDVIFRGPDR
IPSIVASSVT PGVVTAFAEK RVGGGDPGAL SNTNDIITRT SRDGGITWDT
ELNLTEQINV SDEFDFSDPR PIYD--
```

Za zapis proteina FASTA je izuzetno ne praktičan ako korisnika zanima sekundarna struktura, s obzirom da se njime može zadati jedino primarna struktura (sekvenca aminokiselina). U tom slučaju za zapis proteina koristi se PDB (eng. *Protein DataBank*) format.

PDB format izuzetno je opširan tekstualni format, a služi za opis građe i svojstava proteina. On u sebi sadrži trodimenzionalne pozicije atoma, a osim toga još sadrži i podatke o njihovoj međusobnoj povezanosti, sekundarnoj građi, kristalnoj građi itd. S obzirom

proteini mogu biti složeni i od više tisuća atoma, datoteke u PDB formatu mogu biti izuzetno velike i spore za učitavanje (s obzirom da je riječ o tekstualnom formatu).

Tipični primjer PDB datoteke prikazan je u nastavku:

```
HEADER    EXTRACELLULAR MATRIX                      22-JAN-98   1A3I
TITLE     X-RAY CRYSTALLOGRAPHIC DETERMINATION OF A COLLAGEN-LIKE
TITLE    2 PEPTIDE WITH THE REPEATING SEQUENCE (PRO-PRO-GLY)
...
EXPDTA    X-RAY DIFFRACTION
AUTHOR    R.Z.KRAMER,L.VITAGLIANO,J.BELLA,R.BERISIO,L.MAZZARELLA,
AUTHOR   2 B.BRODSKY,A.ZAGARI,H.M.BERMAN
...
REMARK 350 BIOMOLECULE: 1
REMARK 350 APPLY THE FOLLOWING TO CHAINS: A, B, C
REMARK 350   BIOMT1   1  1.000000  0.000000  0.000000        0.00000
REMARK 350   BIOMT2   1  0.000000  1.000000  0.000000        0.00000
...
SEQRES   1 A    9  PRO PRO GLY PRO PRO GLY PRO PRO GLY
SEQRES   1 B    6  PRO PRO GLY PRO PRO GLY
SEQRES   1 C    6  PRO PRO GLY PRO PRO GLY
...
ATOM      1  N   PRO A   1       8.316  21.206  21.530  1.00 17.44    N
ATOM      2  CA  PRO A   1       7.608  20.729  20.336  1.00 17.44    C
ATOM      3  C   PRO A   1       8.487  20.707  19.092  1.00 17.44    C
ATOM      4  O   PRO A   1       9.466  21.457  19.005  1.00 17.44    O
ATOM      5  CB  PRO A   1       6.460  21.723  20.211  1.00 22.26    C
...
HETATM  130  C   ACY   401       3.682  22.541  11.236  1.00 21.19    C
HETATM  131  O   ACY   401       2.807  23.097  10.553  1.00 21.19    O
HETATM  132  OXT ACY   401       4.306  23.101  12.291  1.00 21.19    O
```

Zapis pod oznakom ATOM sadrži koordinate atoma, kojih poplipeptidnom lancu pripada te naravno naziv elementa. Postoje još i drugi podatci, no ovi su najvažnije za opisivanje strukture proteina. Oznakom HETATM opisuju se hetero-atomi, tj. atomi koji nisu sastavni dio proteina, no javljaju se npr. u njegovoj kristalnoj strukturi. SEQRES zapis opisuju sekvence aminokiselina u pojedinom lancu. Oznaka REMARK obični sadrži dodatne upute i komentare (neke od njih standardizirane su i oznakama). Zapisi HEADER, TITLE i AUTHOR sadržavaju informacije o istraživaču/ima koji su kreirali promatrani dokument.

Kao što je vidljivo iz navedenog, pojedini formati za zapis sekvenci i proteinske građe izuzetno se razlikuju. Takvo što naposljetku dovodi do manji problema, ali velikog gubitka vremena koje se troši na konverzije formata umjesto na istraživački rad. Mnogi bioinformatičari rješenje ovom problemu vide u korištenju XML tehnologije, ili točnije u SBML (eng. *System Biology Markup Language*) formatu. Prednost ovog formata (osim što nije obična tekstualna datoteka) jest ta što se njime mogu opisati različite informacije i

procesi (metaboličke reakcije, regulatorne mreže, zarazne bolesti). Struktura i opis ovog formata izlaze iz obima ovoga rada te je on ovdje samo referentno spomenut.

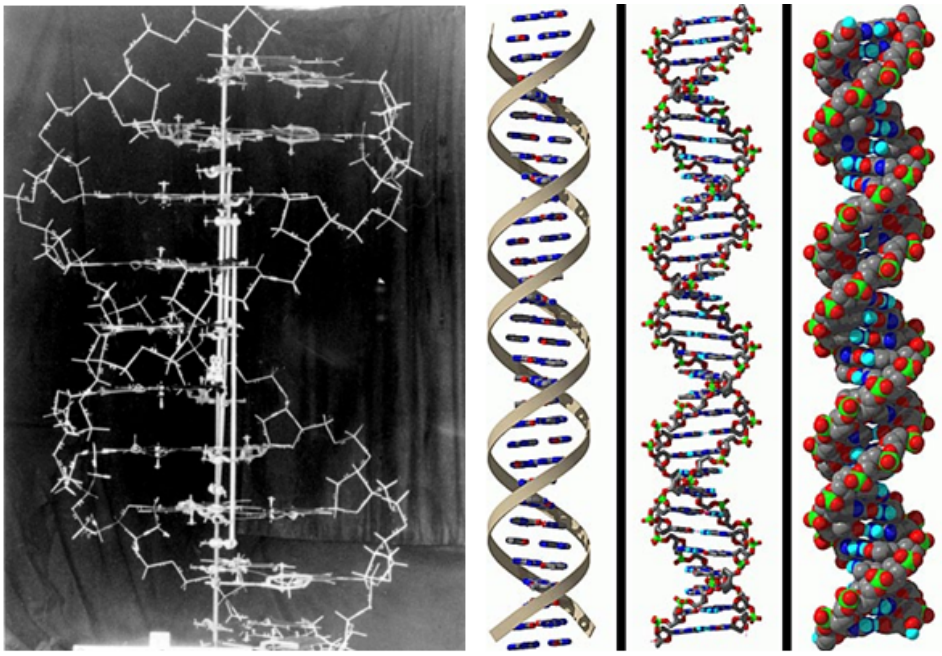
Svi navedeni formati (ali i mnogi drugi) podržani su od stane raznih baza podataka. Nezavisno o samo bazi, korisnicima je omogućeno eksportiranje u formatu koji njima najviše odgovara. Neke od najvažnijih biotehnoloških baza podataka su:

- Ensembl – ljudski genom,
- GenBank/DDBJ/EMBL – nukleotidne sekvence,
- NR, PDB – proteini,
- OMIM – genetske bolesti,
- …

4.2. MOLEKULARNI MODELI I MOLEKULARNA GRAFIKA

Količina (i veličina) mikrobioloških informacija jest prevelika i prekomplicirana da bi se analizirala iz tekstualnih datoteka koje su navede u prethodnom poglavlju. Stoga je ovo poglavlje posvećeno vizualizaciji tih podataka molekularnim modelima. Molekularni modeli proizvod su molekularnog modeliranja – struke koja objedinjuje teoretske i praktične metode raznih područja u svrhu izgrađivanja modela koji vjerodostojno prikazuju neka svojstva određene molekule. Dakle, oni informacije o molekuli prikazuju u grafičkom obliku, i to uglavnom ekstrakcijom relevantnih obilježja iz prethodno navedenih formata. Koriste se u područjima računalne kemije i biologije te fizici materijala u svrhu istraživanja molekularnih i atomskih sustava koji mogu biti jednostavni kemijski sustavi, složene biološke molekule ili smjese materijala (npr. slitine).

Molekularni model jest geometrijska ili topološka reprezentacija molekule ili skupa molekula u krutom, trodimenzionalnom obliku. Njime se vizualiziraju razna svojstva, od relativnih odnosa veličina, prostorne raspodjele atoma do kemijskih veza te promjena u strukturi koje mogu nastati njihovom modifikacijom. Najčešća praktična primjena molekularnih modela jest testiranje hipoteza te korištenje u pedagoške svrhe. Na Slika 13Slika 13 prikazan je jedan od najvažnijih molekularnih modela – model DNK za čije su otkriće i objašnjenje Watson i Crick dobili Nobelovu nagradu 1962. godine.



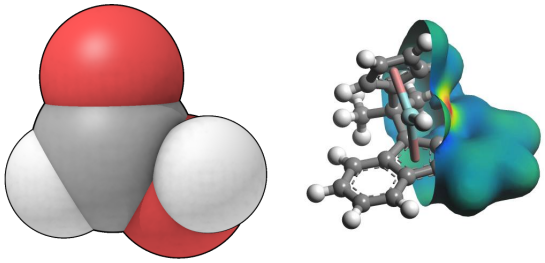
Slika 13. Lijevo: Watsonov i Crickov model DNA. Desno: Trodimenzionalni prikazi DNA pomoću računala.

Molekularni modeli imaju jedan ogromni nedostatak – izuzetno su nepraktični za uporabu (i izradu) kako su u pitanju velike molekule, bilo da se radi o DNA ili o složenim proteinima. Pojavom računalne grafike ovaj je problem postao (donekle) rješiv. Time je započelo doba molekularne grafike.

Molekularna grafika jest disciplina koja proučava molekule i njihova svojstva na osnovi njihove grafičke reprezentacije. Njezin se razvoj temelji na računalnoj grafici i njezinim mogućnostima te na grafičkom dizajnu. Razvoj strukturalne biologije iziskivao je pojavu molekularne grafike već zbog jednostavne činjenice što je bilo potrebno prikazati molekule s tisuću i više atoma, što fizičkim modelom jednostavno nije bilo praktično.

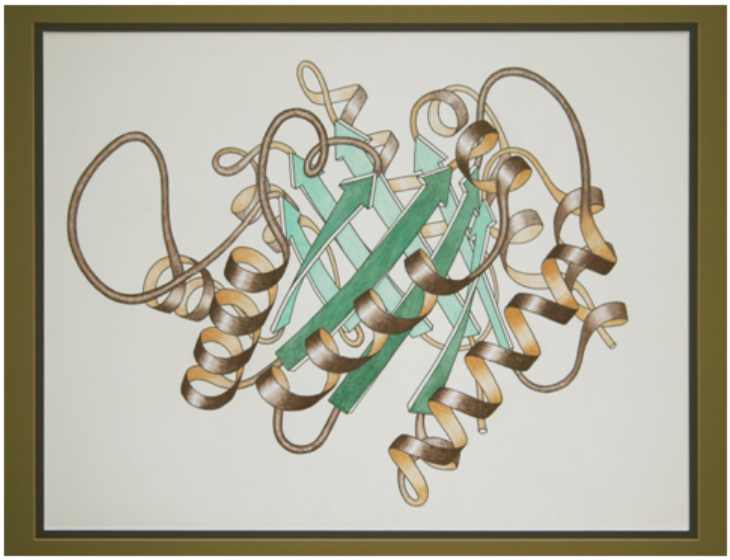
Jednostavne, relativno male molekule i strukture prikazuju se tzv. volumnim modelom (dvodimenzionalni modeli se danas relativno rijetko koriste). U takvom modelu svaki je atom prikazan jednom kuglom određenog (relativnog) radijusa te određene boje. Radijus je tzv. van der Waalsov radijus, koji zapravo označava kuglu koja predstavlja elektronski oblak atoma. Boje su naspram ovog znanstvenog objašnjenja jednostavno konvencija (npr. bijela za vodik, crvena za kisik, crna za ugljik itd.). Veze između atoma jedne molekule se ili prikazuju štapićima ili se ne prikazju – kugle se jednostavno pripajaju jedna na drugu.

Nadogradnjom ovog modela prikazuju se i izopovršine molekula, koja mogu označavati elektrostatski potencijal ili neko drugo svojstvo. Na Slika 14. prikazani su primjeri volumnog modela te izopovršine.



Slika 14. a) Volumni modela mravlje kiseline. b) Primjer izopovršine elektrostatskog potencijala neke molekule.

Pri prikazu složenih molekula, poput proteina koji se nalaze u stanicama živih bića, ovaj pristup ne daje dovoljno pregledan rezultat. 1980. godine biokemičarka Jane Shelby Richardson razvila je vrpčasti dijagram, koji vjerodostojno prikazuje strukturu molekule i njezina najvažnija (strukturalna obilježja), dok samo atomsku građu gotovo potpuno zanemaruje. Slika 15. prikazuje prvi vrpčasti dijagram.

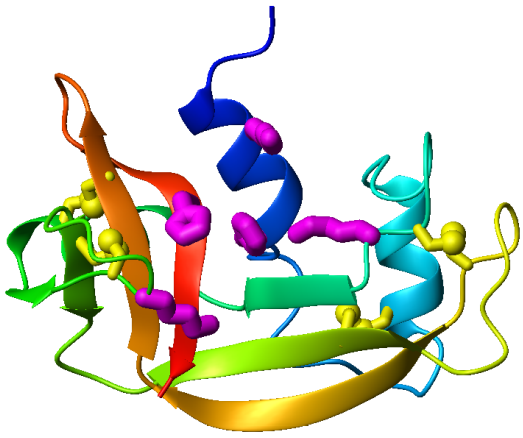


Slika 15. Vrpčasti prikaz trizeofosfat izomeraze, prvi vrpčasti model rukom nacrtan od strane J. S. Richardson

Osnovna kreiranja vrpčastog modela svode se na zamjenu kompleksnih struktura i atoma jednostavnim geometrijskim oblicima. Ti postupci sažeti su u Tabela 2. Primjer učinkovitosti ovog modela dan je na Slika 16. koja prikazuje ribonukleazu-A. Ribonukleaza-A broji preko 3000 atoma.

Tabela 2. Pravila konstruiranja vrpčastog modela molekule iz sekundarne strukture

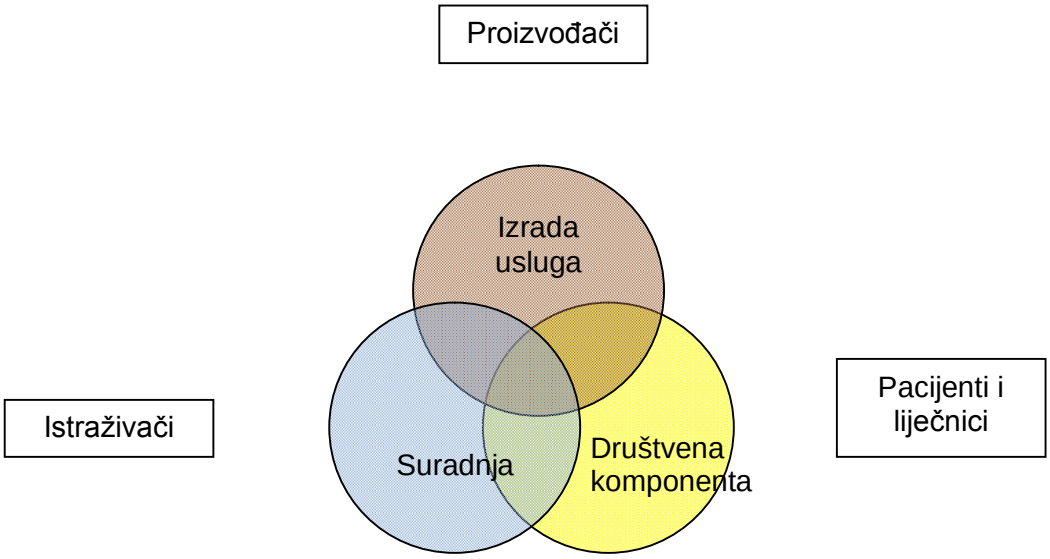
Sekundarna struktura	**Zamjenski Oblik**
α - heliks	Cilindrična, spiralna vrpca, pri čemu vrpca približno slijedi ravninu peptida.
β - niz	Strijelice, debljine obično jednake ¼ dužine, usmjerene i zavrnute od amino prema karboksilnoj skupini. Slično pravilo vrijedi i za beta plohe.
Petlje i razno	**Zamjenski oblik**
Jednostruke petlje	Zaobljene krivulje, podebljane u prednjem dijelu prikazana.
Veze između heliks zavojnica i nizova	Zaobljena krivulja koja postupno prelazi u tanku vrpcu.
Preostala svojstva	**Zamjenski oblik**
Usmjerenje polipeptida	Malene strelice ili slova na oba kraja polipeptida. U slučaju beta nizova ovo nije potrebno s obzirom da je usmjerenje već označeno. Također se može koristiti i stepenasta promjena boje.
Disulfidne veze	Označavaju se SS ili „cik-cak“ simbolom.
Skupine inihibitora	Štapići i kuglice.
Metali	Sfere raznih boja i veličina.



Slika 16. Vrpčasti model ribonukleaza-A.

5. PROJEKT *ACTIV8*

Projekt *Activ8* jest projekt čiji je cilj razvoj sustava i usluga u vidu 3D mrežnog portala za zdravstvenu njegu i pružanje usluga za osnovna biološka i mikrobiološka istraživanja. Zašto je on započet? Zdravstvo je u svojoj osnovi industrija znanja - time je kolaboracija između zaposlenika (liječnika), istraživača i korisnika (pacijenata) izuzetno važna u ovom sustavu. Potreba za razmjenom znanja, mobilnošću i efikasnošću do sada je bila relativno zanemarena iako je ona izuzetno velika u ovome području. Glavni cilj ovog projekta jest iskorištavanje svih postojećih potencijala koje posjeduju liječnici i istraživači na svih razinama i to omogućavanjem trenutnog, nesmetanog pristupa znanju bilo da je se radi o osnovnim biološkim procesima ili pak istraživačkim, dijagnostičkim ili prevencijskim problemima. Ideja je ostvariti povezivanje svih korisnika sustava u jedinstvenu mrežu koja pruža usluge pristupa informaciji i znanju. Na Slika 17. prikazan je dijagram kolaboracije pojedinih elemenata sustava.



Slika 17. Sudionici projekta Activ8.

Kako ostvariti navedenu ideju? U okviru projekta *Activ8* doći do povezivanja više tehnoloških sadržaja u jednu jedinstvenu cjelinu. To uključuje:

- igrače konzole (simulatori negativnog okruženja i pojava),
- medicinske mjerne uređaje raznih namjena,
- mobilne komunikacijske uređaje,

- računala.

Osnovna namjena igraćih konzola jest simuliranje situacija u kojima kod pacijenta dolaze do izražaja simptomi njegove bolesti. Najjednostavniji primjer jesu kondicijski treninzi i testovi (u vidu primjerice računalne igre na Nintendo Wii konzoli) za oboljele od srčanih bolesti, bolesti dišnih putova i sl. Uloga medicinskih uređaja je mjerenje određenih pacijentovih atributa (srčani tlak, bilo, znojenje, volumen udisanog zraka, temperatura, šećer i dr.).
Kako su bežične informacijske prijenosne tehnologije danas izuzetno jeftine (najbolji primjer jest *bluetooth*) njihova ugradnja u spomenute medicinske uređaje ne iziskuje relativno velike troškove. S takvih medicinskih uređaja tada nije problem prenijeti podatke na razne mobilne uređaje zbog jednostavne činjenice što većina njih posjeduje barem neki oblik bežične tehnologije. Naglasak na mobilnosti jest zbog toga što ih korisnici imaju uza sebe gotovo u svakom trenutku – bilo gdje, bilo kada.

Naglasak na bežičnoj komunikacijskoj sposobnosti nije samo zbog komunikacije s medicinskim uređajima; ona je važna i za daljnji prijenos podataka putem Interneta. Naime, ideja jest da se izmjere vrijednosti s medicinskih uređaja prenesu do baze podataka (koja se nalazi na nekom vanjskom poslužitelju) te da se ti podaci naknadno vizualiziraju i obrađuju. U taj računalni sustav nadalje su uključeni liječnici koji nadziru podatke, dok ih i sam sustav analizira (do neke mjere) te dojavljuje moguće medicinske nepravilnosti. Time se uvelike smanjuje vrijeme komunikacije pacijenata i liječnika, vrijeme dijagnosticiranja te vrijeme potrebno za poduzimanje potrebnih prevencijskih mjera.

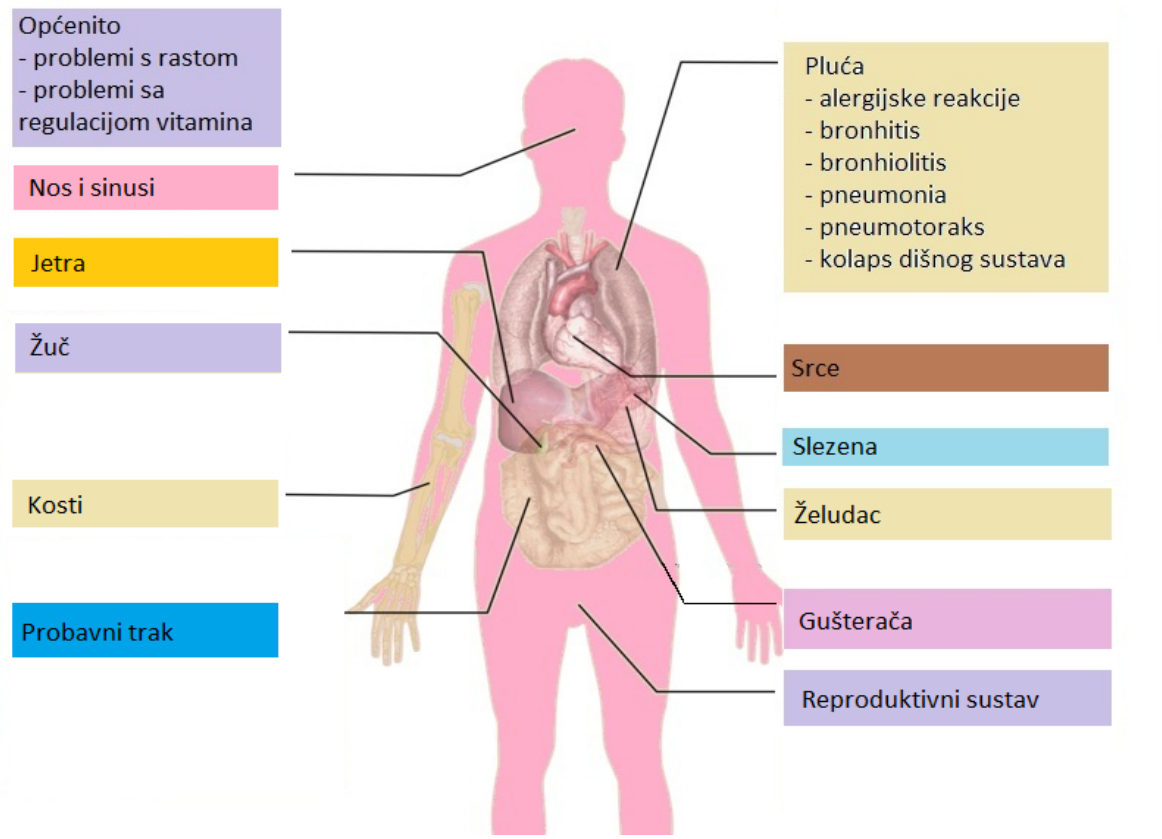
U navedeni računalni sustav također su povezni i istraživači koji imaju pristup većem broju dodatni usluga kojima je cilj olakšati svakodnevnu kolaboraciju te istraživački rad. Više o tome biti će opisano u idućim poglavljima.

5.1. CISTIČNA FIBROZA

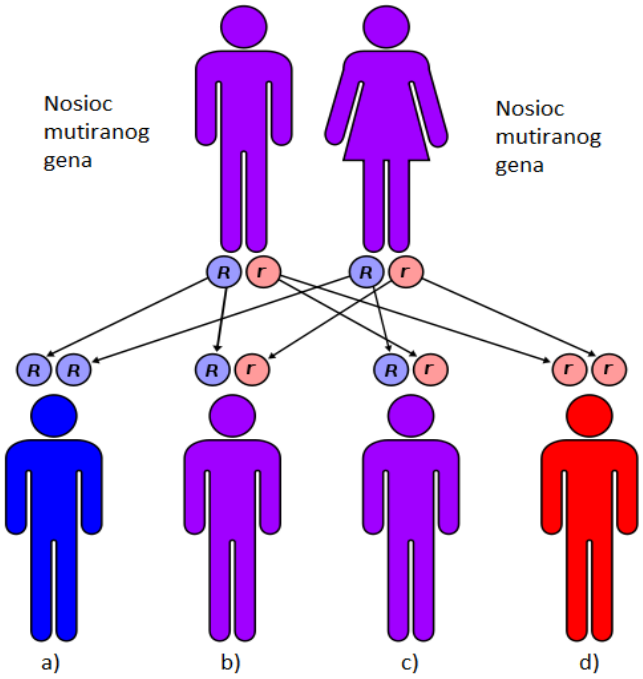
Kako je navedeni projekt još uvijek u prvoj fazi svojeg razvoja (započeo je krajem 2009. godine), navedeni se planovi naravno nisu mogli do sada ostvariti. Stoga je primarni fokus u ovom trenutku da se izgradi rudimentarni sustav s osnovnim mogućnosti za sve vrste korisnika, ali s fokusom na cističnu fibrozu.

Cističan fibroza jest jedna od najčešćih, genetski recesivnih, nasljednih bolesti u bijelaca. To je bolest s velikim brojem simptoma, a najteži slučajevi se otkrivaju već u novorođenčadi ili ranoj dječjoj dobi. Zahvaća veliki broj organa (Slika 18.), a u tipičnim slučajevima su najteže pogođena pluća i gušterača - žlijezda odgovorna za probavu hrane. U većini slučajeva uzrokuje smrt u srednjoj dobi. Bolest je otkrivena 30-tih godina prošlog stoljeća.

Cistična fibroza uzrokovana je mutacijom gena koji proizvodi CTFR (eng. *Cystic fibrosis conductance regulator*) protein. Ovaj protein potreban je za reguliranje sastojaka znoja, probavnih sokova te mukusa (sluzi u dišnom sustavu čija je uloga apsorpcija stranih čestica). Većina ljudi posjeduje dvije kopije ovog gena, te je za pravilan rad dovoljan jedna. Kao što je već rečeno, cistična fibroza jest bolest uglavnom „bijelog“ stanovništva – 1 od 25 osoba europskog podrijetla u sebi nosi jedan gen koji uzrokuje cističnu fibrozu. Slika 19. prikazuje u kojem slučaju dolazi do pojave cistične fibroze u potomstvu.



Slika 18. Posljedice cistične fibroze.

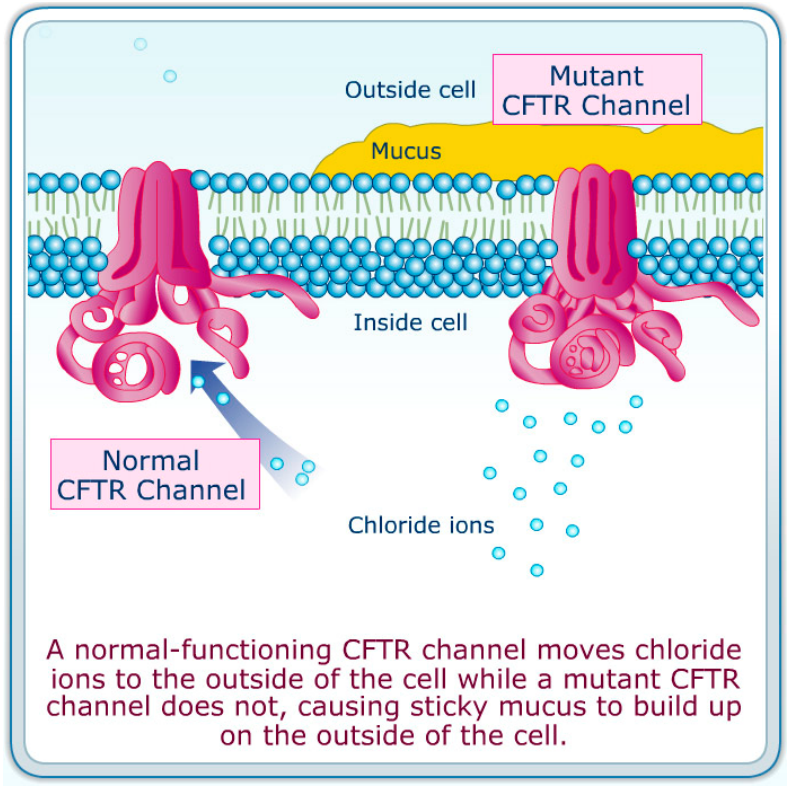


Slika 19. Mogućnosti obolijevanja od cistične fibroze u slučaju da su oba roditelja nosioci jednog, recesivnog mutiranog CTFR gena. a) Nije nosilac mutiranog gena (25%), b) i c) nosioci jednog mutiranog gena (ukupno 50%), d) oba gena mutirana – obolijevanje od cistične fibroze (25%).

Kao što je već rečeno, glavni uzrok cistične fibroze jest mutacija gena koji proizvodi CTFR. CTFR gen nalazi se u 7. kromosomu, na poziciji q31.2. Najčešća mutacija (postoji ih oko 1400) tog gena jest gubitak tri nukleotida što se, nakon procesa transkripcije, očituje kao gubitak aminokiseline fenilalanin na 508. poziciji proteina. Zbog tog gubitka, postupak pregibanja proteina je promijenjen, te on više ne može obavljati svoju funkciju. Ova mutacija se često označava kao Δ508, pri čemu (Δ) označava gubitak.

CTFR se ugrađuje u vanjsku staničnu membranu u znojnim žlijezdama, plućima, gušterači te drugim organima. Ispravan CTFR protein djeluje poput prijenosnog kanala između unutrašnjosti stanice te okolnih fluida. Njegova glavna uloga jest reguliranje kretanja halogena (najviše klorovi ioni – Cl^-) iz unutrašnjosti stanice prema van (obrnuto u slučaju znojnih žlijezda). Kada je riječ o mutiranom CTFR proteinu, on zapravo djeluje poput čepa te ne dopušta reguliranje količine halogena unutar stanice. Posljedice toga su mnogobrojne: nemogućnost stanične osmoze zbog negativnog naboja unutar stanice uzrokovanog Cl^- ionima, ulazak Na^+ iona što čime se stvara sol unutar stanice koja se inače izlučuje znojem,

nemogućnost stvaranja određenih supstanci potrebnih imunološkom sustavu (npr. cijanosulfidnih aniona $OSCN^-$). Shematski prikaz djelovanja ispravnog i mutiranog CTFR kanal prikazan je na Slika 20.



Slika 20. Shematski prikaz djelovanja ispravnog i mutiranog CTFR proteina

5.2. KORISNICI I BUDUĆNOST

U okviru ovog projekta razlikuju se tri tipa korisnika:

- pacijenti (oboljeli od cistične fibroze),
- liječnici,
- istraživači.

Svaki od navedenih tipova korisnika imat će pristup vlastitom interaktivnom okruženju putem web portala. Svaka vrsta okruženja posebno je prilagođena vrsti korisnika, s određeni brojem usluga te 3D okruženjem. Uslugama se može pristupiti izravno putem izbornika ili klikom na reprezentativni objekt u 3D okruženju. Na Slika 21. prikazana su osnovna okruženja.

a) Okruženje namijenjeno istraživačima

b) Okruženje namijenjeno pacijentima

c) Okruženje namijenjeno liječnicima

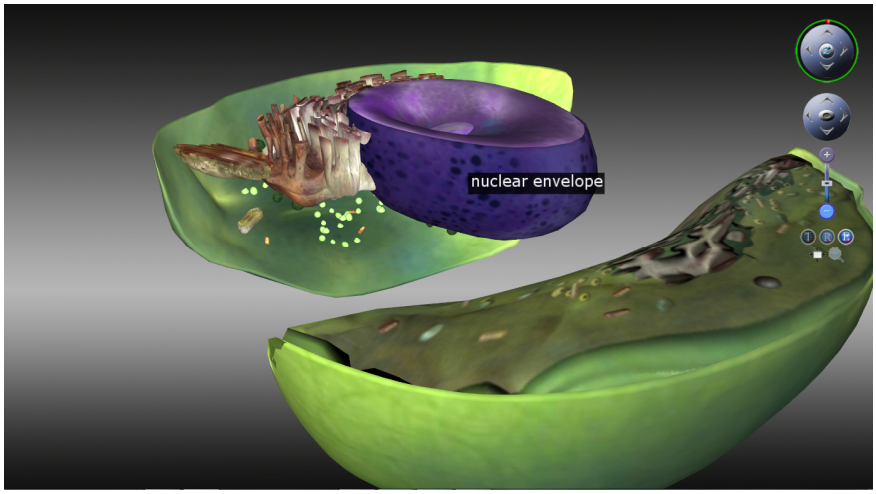
Slika 21. Vrste korisničkih okruženja.

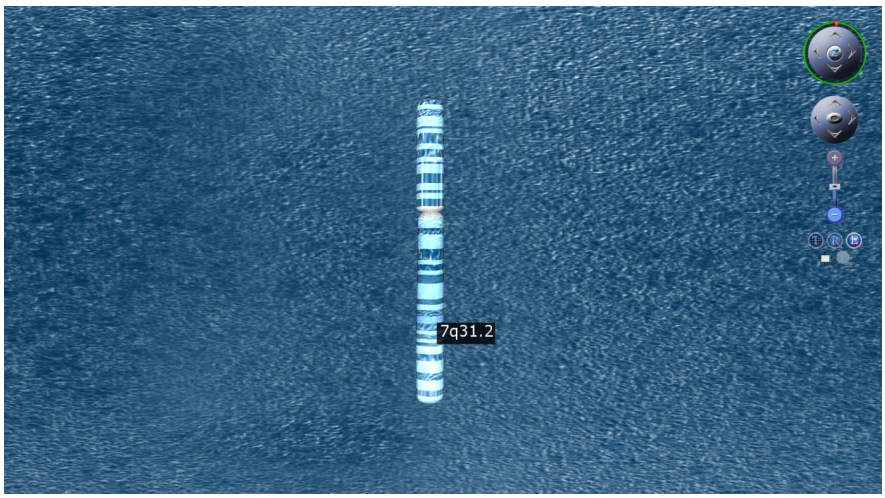
Vrste ponuđenih usluga uključuju:

- Projekti (ne posjeduju pacijenti)
- Članci (na posjeduju pacijenti)
- Zabilješke
- Razgovori putem Skype usluge
- Spremljene stranice
- E-mail poruke
- Kalendar
- Pacijenti (samo liječnici)
- Alati za obradu podataka (samo istraživači)
- RSS usluge

Kao što se vidi iz priloženog glavni fokus jest na što učinkovitijoj komunikaciji među korisnicima sustava. Pacijenti će svoje okruženje primarno koristiti za učitavanje medicinskih podataka te komunikaciju s liječnicima, dok će liječnici imati pristup tim podatcima te mogućnost njihove analize.
Okruženje istraživača jest najopširnije od ponuđenih. Naime, osim navedenih usluga, ono omogućava pristup još jednom dodatnom okruženju – biološkom okruženju. Biološko okruženje specijalno je dizajnirano za potrebe istraživača (prema njihovim zahtjevima). Unutar biološkog okruženja istraživač ima mogućnost učitati veći broj 3D modela – od modela ljudskog tijela, preko modela unutarnjih organa, stanice, kromosoma, sve do razine protein koje generira određeni ekson.

Iako je primarni fokus ovog projekta kreiranje tehnološke podrške oboljelima od cistične fibroze te osobama uključenih u borbu protiv nje, on tu ne završava. Naime, izgradnjom ovakvog sustava postavlja se temelj skupu usluga koja se mogu primijeniti u borbi protiv bilo koje bolesti, bilo gdje u svijetu. I ne samo to, sustav se može proširiti tako se koristi u edukacijske svrhe, ne samo medicine, već bilo koje struke. Svaki bi korisnik posjedovao svoje vlastito, personalizirano okruženje te bi imao pristup podacima i uslugama za koje se pretplatio. Ovo je dugoročni cilj projekta *Activ8* – izgradnja višeslojne edukacijsko-istraživače društvene mreže.





Slika 22. Primjeri bioloških okruženja.

6. PROGRAMSKA IMPLEMENTACIJA

U okviru ovog rada ostvarena je i programska implementacija koja omogućava vizualizaciju bioloških i mikrobioloških podataka te interakciju s korisnikom. U nastavku je da detaljan pregled korištenih tehnologija, uvid u API te primjeri pseudokodova.

6.1. WEBGL I CANVAS ELEMENT

HTML5 jest slijedeća službena verzija HTML-a, koja je trenutno još uvijek u fazi razvoja. Kao što je rečeno u poglavlju 2., jedna od novih mogućnosti biti će i prikazivanje 3D sadržaja. To je omogućeno uvođenjem novog HTML elementa pod nazivom canvas. On omogućuje prikazivanje vizualno dinamičkih 2D i 3D objekata.

Canvas čini dinamički prostor unutar web stranice, a ima definiranu širinu i dužinu. Putem JavaScript koda može se pristupiti definiranoj regiji te, korištenjem većeg skupa funkcija ostvariti dinamički 3D prikaz. Ovaj element jest na nižoj razini nego SVG ako usporedimo njihove API-je. Njegovo programsko sučelje prikazano je u nastavku.

```
interface HTMLCanvasElement : HTMLElement {
    attribute unsigned long width;
    attribute unsigned long height;

    DOMString toDataURL(in optional DOMString type, in any...
args);
    object getContext(in DOMString contextId, in any... args);
};
```

Najvažnija funkcija ovog sučelja jest `getContext(String context).` Naime, putem nje se unutar JavaScript datoteke može ostvariti pristup kontekstu za prikazivanje (eng. *rendering context*). Dva najvažnija konteksta jesu 2D te webgl kontekst. Objekt koji se dohvaća pozivanjem navedene funkcije omogućava pristup funkcijama za 2D operacije nad canvas elementom. U tom slučaju se `canvas` element ponaša doslovce kao platno – korisniku je omoguće izvršavanje raznih manipulacija poput bojanja, pisanja, lijepljenja teksture itd.

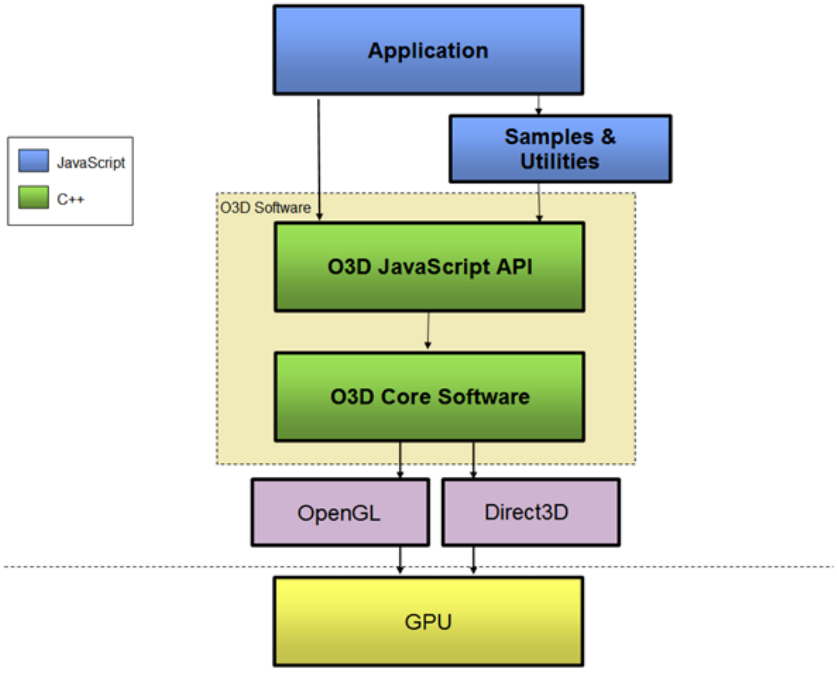
Ono što je međutim izuzetno korisno (i zanimljivo) jest WebGL kontekst. Ovaj je API specificiran od strane Khronos grupe (iste grupe koja je razvila OpenGL) te je najviše

sličan OpenGL ES-u; API-u za grafički prikaz na ugradbenim sustavima. WebGL kontekst (kao objekt) omogućuje uporabu zaista velikog broja funkcija, bilo da se radi o spremnicima, teksturama, sjenčanju, izvorima svijetla itd. WebGL (poput OpenGL-a) jest konačni automat stanja – svaki put kada se pozove funkcija iscrtavanja WebGL prolazi kroz svoja stanja (ovisno o ulaznim parametrima), na posljetku iscrtava sadržaj te ponovno čeka poziv funkcije iscrtavanja.

Dohvaćanjem WebGL konteksta `canvas` element predstavlja i dalje 2D platno, no sada se na njega iscrtava perspektivni (ili ortogonalni) pogled na 3D prostor koji je modeliran WebGL-om.

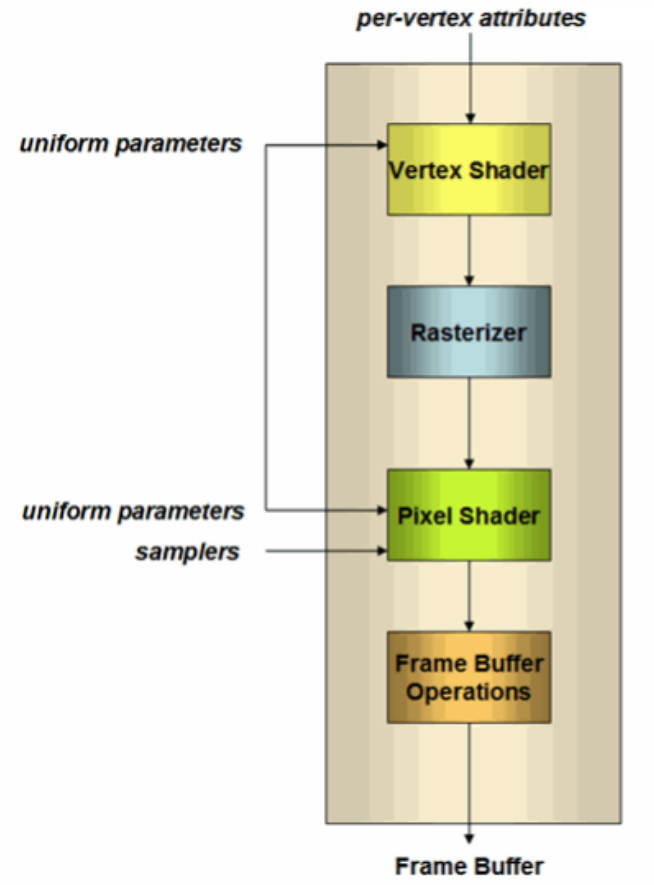
6.2. O3D

O3D jest Googleov besplatni dodatak Internet preglednicima za prikazivanje 3D sadržaja unutar njih. Napisan je u C++-u te je od svoje pojave prije nekoliko godina skupio veliki broj korisnika. Međutim, kao što je rečeno, O3D jest dodatak Internet preglednicima – od korisnika se zahtjeva njegovo instaliranje, što je u nekim okruženjima nemoguće iz npr. sigurnosnih razloga. Nadalje, podržan je samo unutar Windows operacijskog sustava čime se isključuje veći broj potencijalnih korisnika (preko 10%). Položaj O3D-a u hijerarhiji programske podrške tijekom izvršavanja prikazan je na Slika 23**.**



Slika 23. O3D u hijerarhiji programske podrške tijekom izvršavanja.

O3D model procesiranja podataka sličan je rudimentarnom modelu OpenGL-a (2.0 ili više). Taj model temelji se na programibilnom grafičkom sustavu što korisnicima omogućava izradu vlastitih programa za sjenčanje i to uporabom tzv. *vertex* i *pixel shader-a*. Ova tehnologija u suprotnosti je sa sustavom fiksnih funkcija u kojem su algoritmi za proračunavanje bili ugrađeni u programsku potporu poput verzija OpenGL-a prije 2.0. Tim se algoritmima moglo u vrlo maloj mjeri manipulirati.



Slika 24. Glavne komponente O3D sustava za prikazivanje.

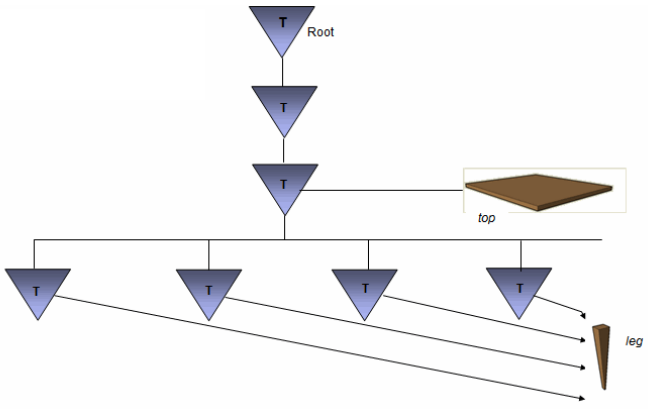
Vertex shader putem definiranog algoritma izračunava koordinate svakog vrha u homogenom prostoru. Uz to, on također izračunava i vrijednosti drugih atributa u danim točkama koji se naknadno interpoliraju (koordinate teksture, boju, normale, težinu itd.). Rasterizator je zadužen za interpoliranje vrijednosti koje mu predaje *vertex shader* te za operacije poput *clippinga.* Izračunati se podatci dalje prosljeđuju *pixel shader-u* koji svakom pikselu određene primitive pridodaje određenu boju. Naposljetku se podatci dodatno obrađuju prije nego se spreme u spremnik okvira (eng. *frame buffer*) – testira se duba, prozirnost itd. Osim navedenih podataka, u proces su uključeni takozvani uniformni

parametri koji se također primjenjuju na primitive, odnosno točke te sampleri koji omogućavaju teksturiranje. To uključuje razne matrice, vektore ili pak decimalne brojeve. Na primjer:

- Matrice
  - projekcija,
  - pogled,
  - model,
  - tekstura,
  - boja…
- Vektori
  - pozicija svijetla,
  - boja svijetla,
  - vektor brzine…
- Brojevi
  - vrijeme,
  - masa…

6.3. POGON ZA OSTVARIVANJE PRIKAZA

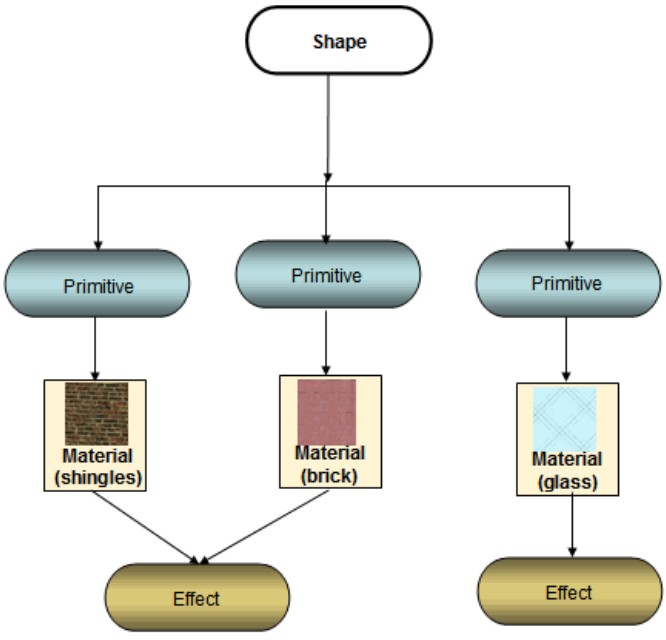
O3D jest API visoke razine – putem njega nije moguće izravno pozvati funkcije grafičkog sustava računala. Temelji se na grafu scene, a također omogućava definiranje parametara poput efekta, materijala i sl. Osnovna gradivna jedinica modela jest primitiva *(*eng. *primitive)*. Primitive su spojene u oblike (eng. *shape*). Primjer na Slika 25. prikazuje kako se pomoću grafa scene i nekoliko jednostavnih oblike može konstruirati složeniji oblik.



Slika 25. Primjer konstruiranja stola pomoću grafa scene.

Svaka primitiva ima barem dva vrlo značajna parametra: materijal i efekt. Efekt u sebi sadrži *pixel* i *vertex shader* koji će se primijeniti nad zadanom primitivom (drugim riječima kod za dva potprograma). Materijal s druge strane sadrži vrijednosti koje se učitavaju u programe za sjenčanje koje sadrži efekt. Ti parametri variraju u zavisnosti o efektu. Primjerice, parametri koji se koriste za konstantno sjenčanje crvene kocke nisu isti kao parametri koji su potrebni za Lambertovo sjenčanje teksturirane kugle. Materijali i efekti su jedinstveni, tj. ne postoje višestruke instance (kopije) istih jer primitive u sebi sadrže samo reference na odgovarajuće objekte. Prikaz odnosa oblika, primitiva, materijala i efekata prikazan je na Slika 26.

Graf scene se najčešće ne konstruira ručno (u kasnijim primjerima pokazano je da i to zna biti korisno u nekim slučajevima). Izgradnja kompleksnih scene poput okruženja prikazanih u prethodnim poglavljima najčešće se obavlja korištenjem specijalnih programa za 3D modeliranje – 3D Studio Max, Maya itd. Nakon oblikovanja scene u nekom od tih programa koristi se programska podrška pod nazivom *COLLADA converter* koja prebacuje originalne formate u .dae format. Dae format predstavlja jednostavan xml dokument koji opisuje graf scene prema definiram standardu. Nakon što se zapis scene prevede u .dae format, koristi je Googleov konverter koji .dae format pretvara u zapis pogodan za čitanje putem O3D-a. Taj zapis predstavlja zapravo komprimirani direktorij koja u sebi sadrži graf scene u JSON formatu (poglavlje 6.4), teksture, spremnik vrhova i dr.

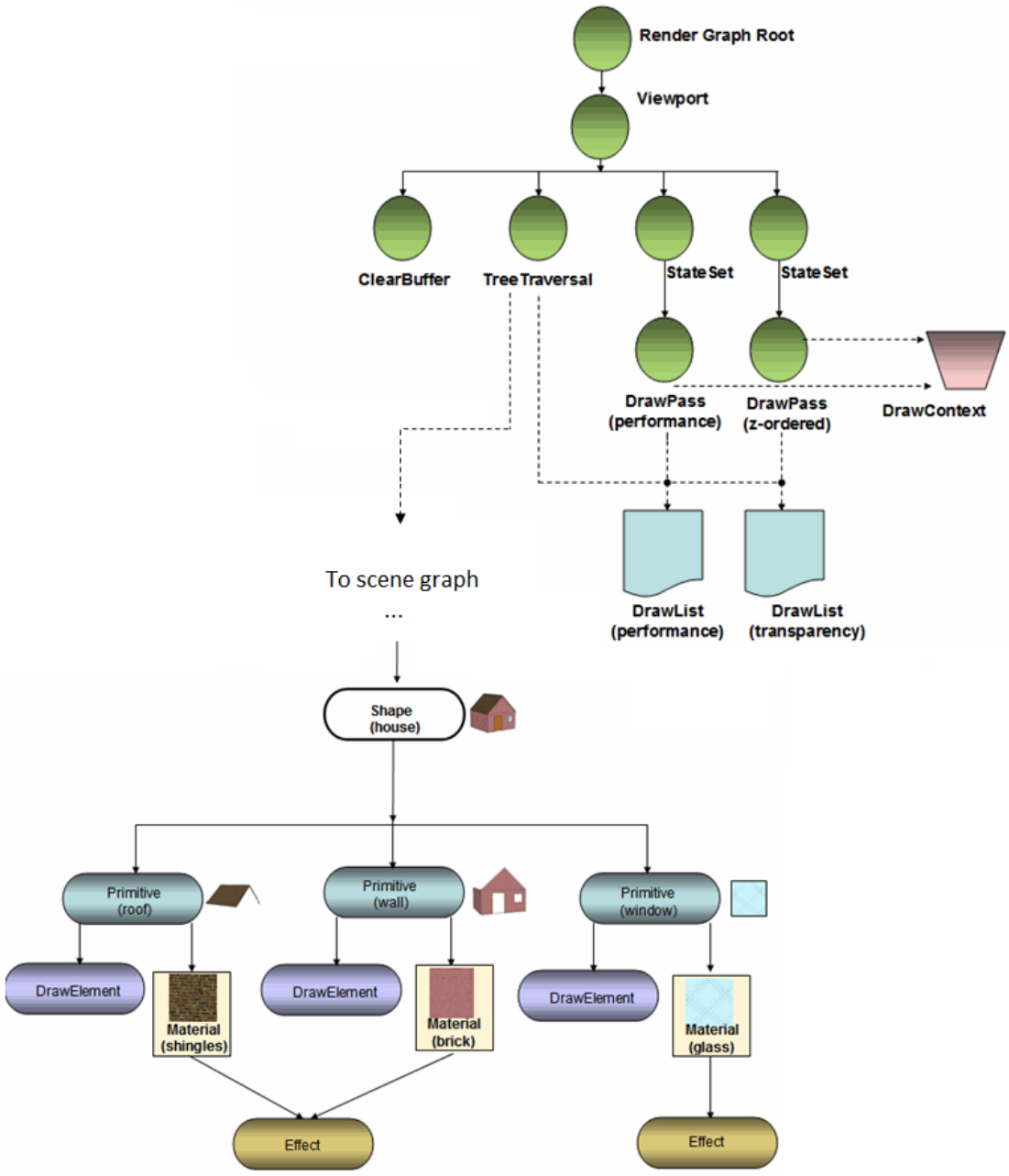


Slika 26. Odnos oblika, primitiva, materijala i efekata.

Nakon što se učita graf scene, on se pripaja kao čvor u graf iscrtavanja. Naime, graf scene samo sadrži njen opis, no on nije dostatan za njeno iscrtavanje. Kako bi se iscrtala zadana scena, primitivi (kao objektu) se dodaje još jedan objekt tipa `DrawElement`. `DrawElement` jest zapravo instrukcija za iscrtavanje, bez nje se ništa ne može iscrtati. Ona također omogućava učinkovito iskorištavanje resursa (npr. jedna primitiva može se iscrtati više puta sa različitim efektima ili materijalima, bez da se ona kopira). `DrawElement` objekti zapisuju se u liste pod nazivom `DrawList`. Postoje dvije takve liste – jedna za netransparentne materijale i druga za transparentne. Slika 27. prikazuje graf iscrtavanja (eng. *render graph*).

Tijek iscrtavanja jest slijedeći:

1. Postavi područje za iscrtavanje (*Viewport*)
2. Očisti trenutni spremnik (npr. ekran)
3. obiđi graf scene te stvori DrawElement objekte
4. provjeri da li je odgovarajuća primitiva vidljiva, ako je dodaj je u odgovarajuću listu za iscrtavanje
5. StateSet objekti postavljaju razne vrijednosti vezane za iscrtavanje (slične OpenGL stanjima)
6. Kontekst iscrtavanja (eng. *draw context*) sadrži matrice pogleda i projekcije. DrawPass objekti izvršavaju iscrtavanje objekata koje se nalaze u listama za iscrtavanje, uračunavajući te matrice. Nakon obilaska cijelog grafa scena se iscrtava na ekran.



Slika 27. Graf iscrtavanja u O3D-u. Isprekidane crte označavaju reference, a pune odnos "roditelj-dijete".

6.4. O3D I WEBGL

Pojavom WebGL-a Google je svibnju 2010. službeno objavio prestanak rada na O3D-dodatku, no nije odustao od ovog projekta. Naime, uz prekid rada na C++ verziji objavio je da će implementirati dani dodatak u JavaScript-u, koji će se temeljiti na WebGL-u i koji će biti otvorenog koda. Sve razloge ove odluke zna samo sam Google, no ono što je bilo očito

i običnom korisniku O3D-a jest nekoliko dobrih posljedica ove odluke. Prelaženjem sa C++-a na JavaScript, korisnici više nisu ograničeni svojim operacijskim sustavom. Nova biblioteka bi također trebala raditi brže s obzirom da nema više komunikacije Internet preglednika s nekim vanjskim dodatkom – sav kod jest u obliku HTML-a i JavaScripta.

Nadalje, u prethodnom poglavlju navedeno je da O3D koristi .o3d datoteke (koji su zapravo komprimirani direktorij) koji u sebi sadrži i teksture scene. Izravna posljedica toga jest da svaka tekstura mora biti kopirana onoliki broj puta koliko scena nju sadrži. Drugim riječima, ako je neka tekstura korištena u *n* scena, ona se mora nalaziti i u toliko .o3d datoteka, bez obzira što se njima pristupa s iste web stranice. Uvođenjem WebGL-a sadržaj .o3d datoteka više ne mora biti kompresiran, već je za opis scene dostatna JSON datoteka koja u sebi sadrži staze do potrebnih tekstura (JSON datoteka jest zapravo samo jedna komponenta ove kompresirane datoteke). Na taj je način omogućena velika ušteda memorijskog prostora, ali i smanjenje mrežnog prometa.

No, postoji još nekoliko razloga zašto je ova odluka bila dobra. Prelaskom na JavaScript modifikacija koda i njegova prilagodba konkretnom problemu postala je moguća - bilo da se radi o dodavanju funkcionalnosti, njezinom odbacivanju ili optimizaciji. Četvrta pozitivna posljedica ove odluke jest što se funkcionalnost O3D-a postala iskoristiva i na mobilni uređajima baziranim na iOS (iPhone, iPad) i Android operacijskom sustavu. S obzirom da je broj korisnika računala u svijetu oko 400 milijuna, a do sada je samo iPhone prodan u više od 70 milijuna, onda je zaista očito koliko raste baza korisnika (a time i prostor za profit).

WebGL implementacija O3D-a jest po svojoj strukturi identična dosada opisanim svojstvima i obilježjima O3D-a. Dakle, sučelje ove programske podrške ostalo je i dalje nepromijenjeno. Unatoč svim ovim dobrim značajkama, WebGL implementacija (kao i originalni O3D) imaju jedan mali „nedostatak“ – koriste se unutar JavaScript funkcija web stranice. Ne može se tvrditi da je to nedostatak u stvarnom smislu riječi, no JavaScript datoteke postaju izuzetno nepregledne i teške za održavanje kada postanu relativno velike. Glavni razlog tomu jest što ne postoji razvojno okruženje koji omogućava jednostavno upravljanje takvim radnjama (poput npr. Eclips okruženja za Javu ili Visual Studio okruženja za C#).

Tomu se nedostatku može doskočiti jednim malim trikom - korištenjem *Google Web ToolKit-a* (GWT). GWT jest skup alata koji služe kao pomoć inženjerima programske podrške za razvoj i održavanje kompleksnih *front-end* JavaScript aplikacija putem Java programskog jezika. Naime, osnovna ideja jest da se sav potreban kod za izradu neke web stranice napiše u Javi te naknadno, putem GWT, samo prevede u JavaScript i HTML. Tom se rješenju upravo pristupilo i u ovom slučaju te je cijela biblioteka prevedena u Java programski jezik. Time se u konačnici uštedila velika količina vremena tijekom kreiranja aplikacije, ali i njezinom održavanja.

Tabela 3. Ocjene pojedinih tehnologija u okviru razvoja projekta Activ8.

API/Pogon	**Kvaliteta prikaza i vizualizacije**	**Programiranje**	**Interakcija**	**3D modeliranje**	**Ukupno**
O3D (Google)	10	9	9	9	37
Unity	10	5	10	10	35
Away3D	8	5	9	9	31
Java 3D	8	10	9	9	36
JOGL	10	9	9	7	35
LWJGL	7	7	9	7	30
JME	10	10	9	9	38
3D MaX	NA	NA	NA	10	10
Maya	NA	NA	NA	10	10

Nakon svega navedenog i dalje ostaje pitanje zašto baš ova tehnologija, a ne neka druga? Odgovor je dan u Tabela 4. koja se nalazi u privitku. Iako O3D nema najviše bodova prema ovim kriterijima, ipak je on odabran zbog mogućnosti iskorištavanja na mobilnoj platformi.

6.5. MODIFIKACIJE WEBGL IMPLEMENTACIJE O3D-A

O3D je s programskog stajališta doslovce samo programska podrška za iscrtavanje sadržaja. Kao takav on ne omogućuje zahtijevanu korisničke interakcije, niti približno nešto što bi se moglo nazvati pogon za simulaciju ili igranje. S toga je bilo potrebno određene module dodati u svrhu proširenja funkcionalnosti, a druge modificirati u svrhu optimizacije.

6.5.1. Optimizacija postojećih modula

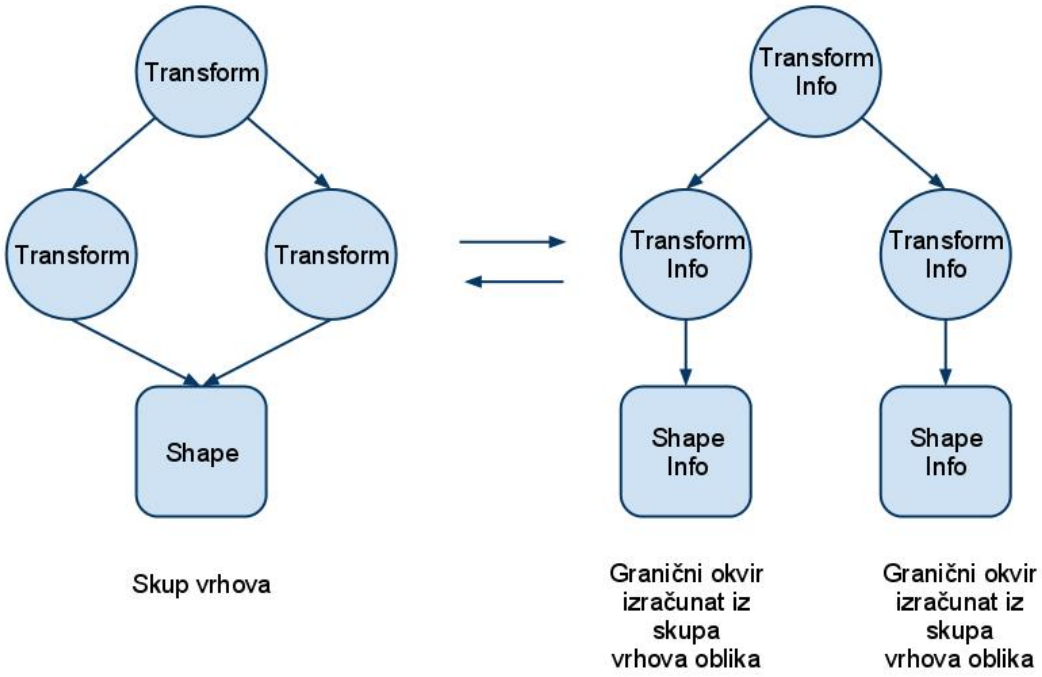
Kako bi se promatrani pogon prilagodio problematici, bilo je potrebno provesti određene optimizacijske promjene, bilo da se radi o količini memorije, procesorskom vremenu ili mrežnom prometu.

Selekcija objekta i detekcija kolizije

`TransformInfo` objekt služi za spremanje podataka o pojedinom (transformacijskom) čvoru u grafu scene. Generira se svaki puta iznova kada dođe do promjene u sceni i sadrži objekt pod nazivom `ShapeInfo`, koje nosi informacije o oblicima. Glavna namjena jest izračunavanje graničnih okvira (oblika ili skupa oblika koji se nalaze pod nekim čvorom stabla). Kada se stvore ti objekti oni zapravo predstavljaju stvarno stanje scene. Naime, unutar te stablaste strukture ne postoje petlje, i ona sadrži upravo onoliko oblika koliko nalazi u sceni – ako je jedan oblik kopiran tisuću puta, tada postoji i tisuću instanci `ShapeInfo` objekta u `TransformInfo` stablu. Vizualizaciju te konverzije prikazuje Slika 28.

Glavna namjena `TransformInfo` grafa jest odgovoriti na pitanje da li je neka zraka (eng. *ray*) došla u doticaj s nekim oblikom. Obilaskom grafa testira se da li je neka zraka presjekla neki `TransformInfo` čvor (testiranje se provodi provjerom intersekcije s graničnim okvirom). Ako jest, onda se tada se taj čvor pretražuje dalje u dubinu sve dok se ne pronađe najbliža intersekcija sa nekim oblikom. Međutim taj postupak, s obzirom na veličinu današnjih računala ne iziskuje previše rada. Problem u službenom algoritmu jest

što se granični okvir izračunava za svaku instancu određenog oblika. S obzirom da se radi o grafu scene, to nije potrebno jer je granični okvir u lokalnim koordinatama oblika, te se zapravo za svaku instancu istog oblika dobivaju isti brojevi. Ako sada razmotrimo primjer vizualizacije nekog proteina koji ima nekoliko tisuća atoma (svi su izvedeni iz istog oblika kugle), tada vidimo da navedeni algoritam radi sa složenošću *O(n)*, iako bi logički trebao raditi sa složenošću *O(1)*.



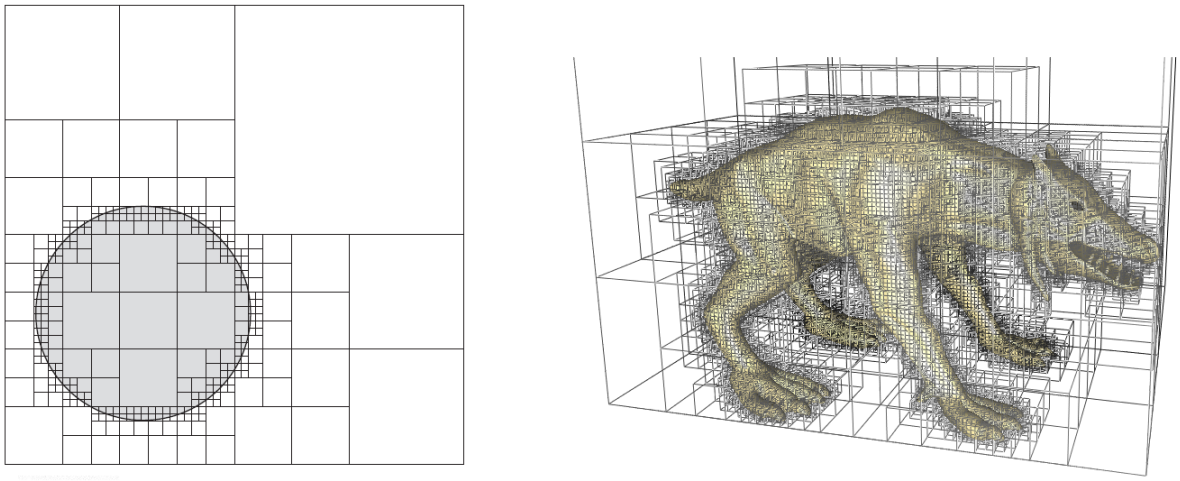
Slika 28. Istovjetnost Transform i TransformInfo grafa.

Upravo zbog ovog nedostatka, navedena je klasa modificirana tako da i `TransformInfo` stablo sadrži upravo onoliko `ShapeInfo` objekata koliko ih sadrži i stvarni graf scene, čime se jako reduciralo vrijeme potrebno za njegovo generiranje.

Ova programska struktura koristi se i kao osnova za detekciju kolizije (kamere) s objektima u sceni. Objašnjenom modifikacijom znatno je smanjeno vrijeme pretraživanja intersekcije, no u slučaju da objekt obiluje jako velikim brojem poligona (reda više stotina tisuća vrhova), ni ona nije dostatna za kretanje u stvarnom vremenu. Naime, i dalje se

pretražuje velik broj čvorova za koje je zapravo moguće i ranije utvrditi da kolizija nije moguća s njima.

U tu svrhu navedena je struktura modificirana implementacijom oktalnog stabla. Oktalno stablo služi za dijeljenje prostora na manje cjeline, koje se zatim odvojeno mogu pretraživati. U najjednostavnije slučaju (koji je bio dostatan za ovaj rad) početni je prostor (kocka) dijeli na osam manjim dijelova. U slučaju da neki od tih dijelova sadrže objekte, prostor se dalje dijeli da isti način, a u slučaju da je prazan on postupak dijeljenja prestaje. Rezultate algoritma za 2D i 3D okruženje prikazuje Slika 29.



Slika 29. Primjeri kvartarnog i oktalnog stabla.

Malo, ali učinkovito

Osim navedenih promjena koje uvelike pridonose ubrzanju rada aplikacije, postoji još nekoliko manjih koje u određenim fazama značajno povećavaju brzinu iscrtavanja. Jedna od njih je modifikacija JSON datoteke koja sadrži opis scene. Naime, nakon prebacivanja iz .dae u JSON format, spremnik vrhova sadrži doslovce sve vrhove scene kao i njihove normale, koordinate teksture itd. u jednom nizu, s pripadajućim spremnikom indeksa. Problem jest u tome što su podatci sortirani po značenju – prvo idu podaci o koordinatama, zatim o normala, nakon toga o teksturi itd. To dovodi do posljedice da se prilikom svakog učitavanja scene (kada se dohvaćaju podatci za pojedini vrh) mora „skakati po spremniku vrhova“, što izuzetno usporava sam postupak učitavanja grafa scene.

Ovaj je problem riješen tako što je u svakoj JSON datoteci proveden prerazmještaj elemenata spremnika vrhova, tako da su sada podatci vezani za pojedini vrh u nizu. Time se omogućilo serijsko čitanje spremnika (WebGL omogućuje čitanje ne samo jednog elementa, već i više uzastopnih odjednom), Što je drastično ubrzalo učitavanje scene. Primjera radi, sceni koja sadržava oko 3000 vrhova prije modifikacije je trebalo oko 40 sekundi za učitavanje, nakon toga nešto manje 4. Dakle, u primjeru navedena je modifikacija izazvala ubrzanje prvog iscrtavanja scene za više od 90%!

Postoji još jedan primjer vrlo učinkovite modifikacije koje ne iziskuje puno truda. Kao što je navedeno u poglavlju 6.4, scena više nije komprimirani direktorij, već se sastoji od više datoteka - opisa scene u obliku JSON datoteke i korištenih tekstura (.png, .jpeg). Istaknuto je da se ovakvim restrukturiranjem postigla nezanemariva ušteda memorijskog prostora i količine mrežnog prometa. No, ovu optimizaciju moguće je provesti i korak dalje.

Promotrimo primjer vrpce korištene u vrpčastom dijagramu proteina. Najizravniji način njezinog prikaza jest kreiranje 3D modela u npr. 3d Studio Max-u. Takav jedan objekt sadrži mnoštvo točaka, i uz bi se moralo generirati više različitih varijanti vrpci, s obzirom da nisu sve jednake dulje. Očito je da se takvim pristupom previše specijalizira programska podrška te on nije dobro rješenje. Drugi način bio bi izračun koordinata vrhova vrpce prilikom učitavanja .PDB datoteke, no ni to ne daje dobre rezultate. Iako programska podrška time postaje za vizualizaciju vrpce postaje neovisna o molekuli koja se želi prikazati, programsko kreiranje takve vrpce i dalje uzima previše vremena.

Rješenje ovog problema jesu programi za sjenčanje. Kao što je već navedeno, postoje dvije različite vrste programa za sjenčanje u sklopu WebGL-a: *vertex shader* i *pixel shader*. Slika 26. Odnos oblika, primitiva, materijala i efekata.Slika 26. (poglavlje 6.3) opisuje odnos oblika, primitiva, materijala i efekata – objekata koje sadrže navedene programe u tekstualnom formatu. Na koji pak način ovi programi ubrzavaju proces izrade i prikazivanja oblika? U prethodnom primjeru spomenuto je generiranje vrpce koja se pomoću programa za sjenčanje može vrlo jednostavno realizirati. Naime, da bi se generirao model vrpce moguće je konstruirati valjak (ovaj postupak nije kompliciran niti računalno zahtjevan), a zatim pomoću *pixel shader-a* „obojiti“ određene njegove točke koje formiraju vrpcu, a ostale ostaviti prozirnima. Na ovaj način postiglo se znatno ubrzanje generiranja oblika vrpce, i to na generički način.

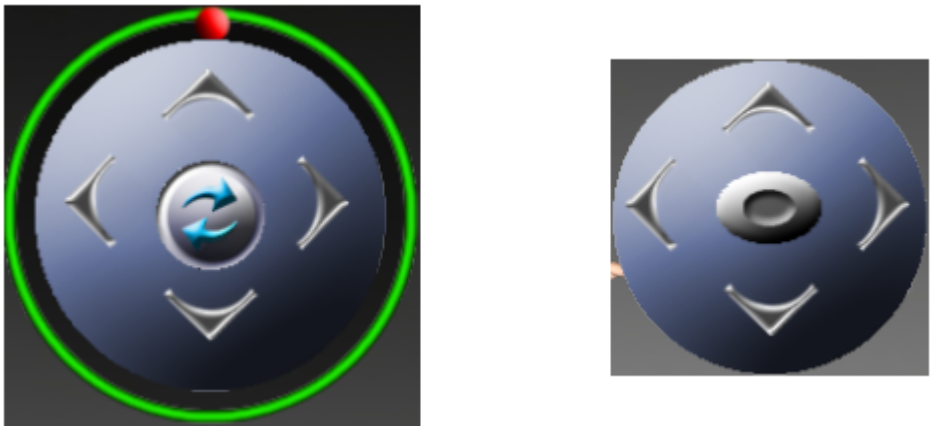
Iako se programi za sjenčanje najčešće koriste za dodavanje efekta materijalima (poput odsjaja ili sjene), oni se mogu iskoristiti i na ovaj način ako nekom objektu nije potrebna složena tekstura, već samo boja. Osim toga, programi za sjenčanje su i sami po sebi izuzetno učinkoviti jer se izvršavaju na grafičkom procesoru uz visok stupanj paralelizacije (veći broj vrhova ili piksela obrađuje se istovremeno) što donosi izrazit skok u performansama ako se koriste u rješavanju ovakvih i sličnih problema. Ovaj pristup ima osim brzine još jednu pozitivnu posljedicu – korištenjem programa za sjenčanje ponekad je (kao u ovom primjeru) moguće izbjeći korištenje teksture, što za dodatno ubrzava iscrtavanja scene, a također dovodi do smanjenja količine zauzete memorije i mrežnog prometa.

6.5.2. Dodavanje novih modula

U prethodnom poglavlju opisano je kako se putem `TransformInfo` grafa prepoznaje selekcija objekta. Međutim, to je sve što O3D vraća – logičku vrijednost *da* ili *ne*. U interaktivnom okruženju gdje je glavni cilj vizualizacija podataka takav rezultat jednostavno nije dovoljan.

Kretanje u O3D nije implementirano – jednostavno zbog činjenice što je on definiran kao pogon za ostvarivanje prikaza te ne sadrži podršku kojom bi se simulirala fizika ili kretanje. Stoga je bilo potrebno dodati nove module u postojeću programsku potporu kako bi se ono ostvarilo. Omogućeno je kretanje kamere naprijed-nazad, rotacija oko njezine 3 osi, te rotacija kamere oko ishodišta grafa scene. Navedene mogućnosti implementirane su modificiranjem matrice pogleda, ali i modifikacijom korijensko čvora grafa scene.

Ono što je u ovom slučaju važno za istaknuti jer korisničko sučelje za kretanje. Osim standardnih kontrola (izravno putem miša i tipkovnice) omoguće je i kretanje korištenje tzv. navigacijskog panela (Slika 30). Navigacijski panel jest zasebni graf iscrtavanja koji se automatski inicijalizira prilikom ulaska u pojedino okruženje. On pripada zasebnom grafu zbog činjenice što se on naknadno iscrtava, nakon grafa scene (nasuprot pozadini scene koja se iscrtava prije). Zbog je također neovisan o trenutnom položaju kamere, a korisnicima omogućuje dodatni način interakcije. Panel se sastoji od dva dijela – sučelje za upravljanje kamerom i sučelje za upravljanje scenom (samo u biološkim okruženjima). Iako je u trenutnoj verziji implementiran kao 2D element, u budućnosti se planira zamijeniti trodimenzionalnim objektom.



Slika 30. Navigacijski panel. a) Scena, b) kamera.

Nakon što je ostvareno kretanje bilo je potrebno implementirati i detekciju kolizije. U prethodnom poglavlju opisano je kako je uvedeno oktalno stablo te na koji način se iskorištavaju podatci zapisani u `TransformInfo` grafu. Implementirano je više verzija detekcije kolizije, od jednostavnijih prema složenijima, ali i računalno zahtjevnijima. Zašto više verzija? Kao prvo, nisu bile poznate karakteristike sustava te u kolikoj će mjeri biti opterećen testiranje kolizije.

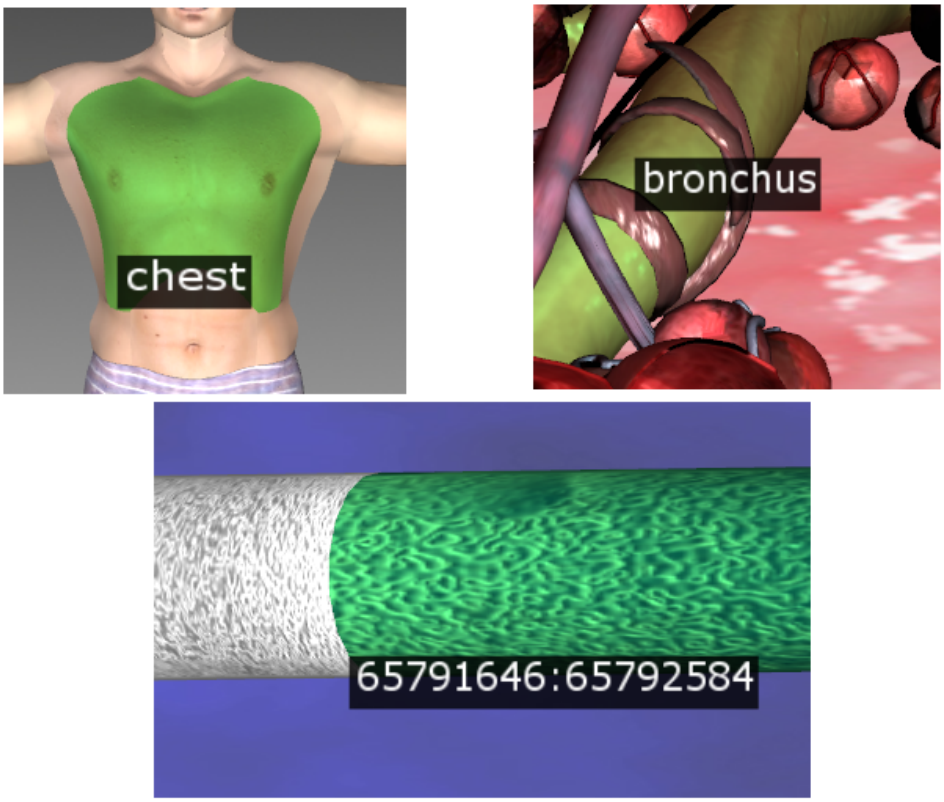
Implementirane su tri verzije detekcije kolizije:

- ograničavanje kretanja kamere,
- detekcija kolizije graničnim sferama (eng. *bounding spheres*),
- detekcija kolizije graničnim okvirima.

Prva od navedenih metoda zapravo nije detekcija kolizije u pravom smislu te riječi. Naime, kretanje kamere se ograničava s obzirom na neki predefinirani volumen ili plohu, te se na taj način simulira kontakt kamere i primjerice zida ili stropa. Takav način „detekcije kolizije“ implementiran je u okruženjima pacijenata, liječnika i istraživača. S obzirom da ta okruženje predstavljaju zatvorene prostore (radne sobe, laboratorije itd.) čija je primarna namjena interakcija s objektima koji pozivaju dostupne usluga (e-mail, kalendar, bilješke...), nije bilo potrebe kreirati napredniji algoritam. implementacijom naprednije detekcije kolizije korisniku bi se samo otežalo jednostavno kretanje od jednog objekta prema drugome, te bi navedena okruženja izgubila svoju primarnu funkcionalnost.

Druga vrsta detekcije kolizije koristi se u okruženju proteina. Razlog tomu jest što se za prikaz proteina koriste najviše sfere, što znači da se granične sfere ne moraju niti izračunavati – nju predstavljaju sami atomi. Testiranje kolizije dvije sfere (objekta i kamere) jest izuzetno jednostavno i svodi da ispitivanje udaljenosti njihovih središta. Treća vrsta kolizije jest idejna jednaka prethodnoj, osim što koristi granične okvire, a ne sfere. Granični okviri u prosjeku bolje obuhvaćaju objekte od sferičnih (izuzev samih sfera i sferičnih oblika). Drugim riječima, detekcija kolizije jest učinkovitija, ali i računalno zahtjevnija, s obzirom da za detekciju kolizije više nisu dostatne dvije vrijednosti (radiju i središte), već ih je potrebno šest (šest ploha kvadra).

Omogućavanjem kretanja i detekcijom kolizije postiže se veća realnost aplikacije. Međutim, s obzirom da se radi i o edukativno-istraživačkoj podršci, bilo je potrebno dodati još jedan manji dodatak – informacijski panel (Slika 31). Informacijski panel služi za prizivanje informacija o selektiranom objektu. U ovom trenutku radi se samo o njegovom imenu, no buduća proširenja će uslijediti (dohvaćanje opisa iz baze podataka i sl.). Informacijski panel dio je grafa iscrtavanja kojem pripada i navigacijski panel. Razlog tome jest jednostavno što se navedeni panel također treba uvijek iscrtavati nakon 3D scene.



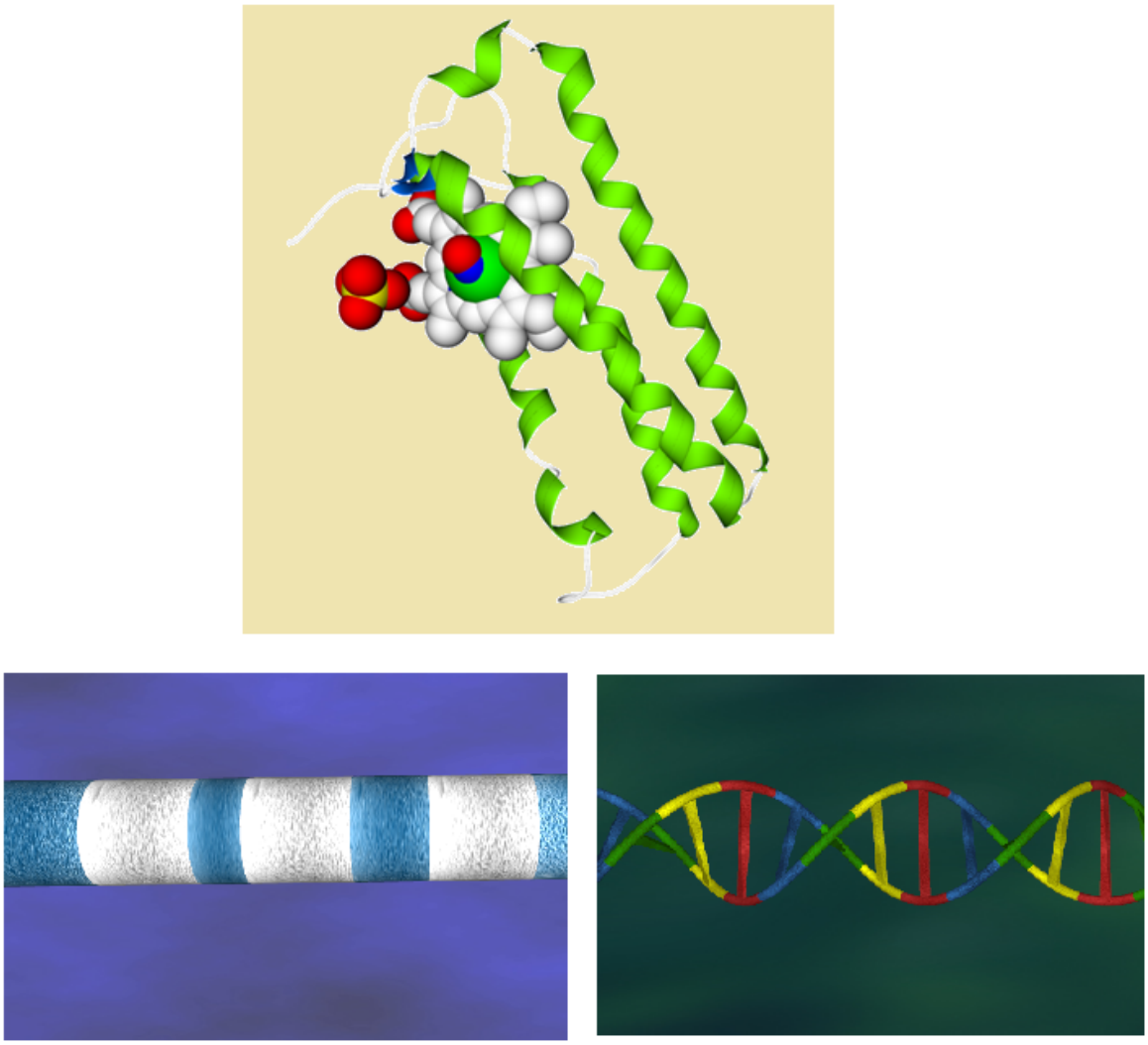
Slika 31. Primjeri informacijskog panela u različitim okruženjima.

Geni, eksoni i proteini

Okruženja gena, eksona i proteini razlikuje se od ostalih okruženja. Naime, ova okruženja su generička – nije ih potrebno posebice modelirati već se graf scene izgrađuje tijekom izvođenja aplikacije, u stvarnom vremenu. Razlog takvom pristupu jest jednostavno količina podataka koja se može prikazati. Ljudski genom sadrži oko 100 000 različitih gena, a gen u prosjeku ima 8.8 eksona (ukupno 880 000 eksona). Uračunavajući i proteine koje ljudsko tijelo proizvodi već se dođe do brojke milijun.

Generički pristup prikazivanju spomenutih elemenata nije previše kompleksan. Glavna ideja jest učitati osnovni gradivni element pojedine strukture (valjak, jedan zavoj alfa-heliksa ili kuglu) te čitanjem iz odgovarajućih datoteka (ili baza) dohvatiti podatke o položaju, veličini, vrsti elementa itd. Slijedeći korak jest teksturiranje objekata, što se vrlo učinkovito postiže grafom scene. Naime, pomoću njega je dostatno učitati svega 6 tekstura za uspješno prikazivanje okruženja gena i eksona. U slučaju proteina čak se ni ne koriste

teksture, već programi za sjenčanje, što dodatno ubrzava iscrtavanje. Slika 32. prikazuje primjer navedenih okruženja.



Slika 32. Korištenje programa za sjenčanje u biološkim okruženjima.

7. DALJNJI RAZVOJ

U okvira projekta *Activ8* postoji mnogo mogućnosti za dodatna proširenja. Jedan slijedećih koraka jest svakako prelazak na mobilnu platformu (poglavlje 6.4.). Postupak prijelaza ne bi trebao iziskivati previše napora ili vremena jer WebGL bi trebao omogućiti izvođenje istog programskog koda kao i računalo. U prethodnoj rečenici koristi se kondicional zbog toga što je WebGL još uvijek razvojnoj fazi. Tek objavljivanjem službene verziji i potvrde od strane proizvođača Internet preglednika da njihovi proizvodi podržavaju u potpunosti WebGL biti će moguće ostvariti potpunu migraciju (u ovom trenutku još postoje problemi s Firefox implementacijom).

Nadalje, potrebno je što više modula za generičko izgrađivanje okruženja (kao što su geni i proteini). Takvi moduli mogli bi se primijeniti za automatsko generiranje okruženja liječnika, pacijenata i istraživača (dakle, sobe ili laboratoriji). Uz to korisnicima bi se pružila i mogućnost personalizacije vlastitog okruženja novim objektima (uz odgovarajuću naknadu), njihov razmještaj i u konačnici izrada vlastitih modela.

Projekt se nadalje može proširiti u vidu tematike koje pokriva – najprije bi fokus sa cistične fibroze proširio na druga medicinska i biomedicinska područja, a izradom odgovarajućih alata i usluga i šire, na druge znanosti. Omogućavanjem pristupa toliko količini informacija, ovaj bi se projekt mogao proširiti s domene e-zdravlja i na e-školstvo. Primjerice, atlas anatomije za medicinski fakultet košta gotovo tisuću kuna, dok bi naknada za pristup ovakvom sustavu bila drastično manja. Uz to, cijeli bi se sustav mogao oblikovati u vidu jedne virtualne, edukativno-istraživačke zajednice.

8. ZAKLJUČAK

Bioinformatika kao znanost novije generacije predstavlja izuzetno intrigantno područje istraživanja. Kolaboracija informatike i biologije napreduje velikom brzinom i otkriva nevjerojatne količine informacija svakodnevno. Da bi se taj korak održao, uvijek su potrebne nove metoda i alati koji mogu pomoći istraživačima u njihovom poslu. Uz područje bioinformatike usko je vezano i područje medicine koje iskorištava otkrića bioinformatičara u svrhu boljitka ljudske rase. I u tom segmentu programska je podrška izuzetno bitna, bilo da se radi o komunikaciji i kolaboraciji liječnika (i pacijenata) ili o sustavima za vizualizaciju, prijenos i analizu biomedicinskih podataka. U tu svrhu WebGL programski API pokazao se izuzetno korisnim jer udovoljava svim postavljenim zahtjevima. O3D kao biblioteka više razine također se pokazala učinkovitom, uz dodatne modifikacije i optimizacije.

Industrija e-zdravlja u moderno je doba prijeko potrebna kako bi se što brže i učinkovitije reagirali na zdravstvene probleme pojedinaca. Stoga razvoj ovog područja valja poduprijeti i novčano i ljudskim resursima kako bi ljudski život bio što kvalitetniji i dugotrajniji.

LITERATURA

[1] *Bioinformatics*, prosinac 2010., http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html

[2] *Sequence Aligment*, prosinac 2010., http://en.wikipedia.org/wiki/Sequence_alignment

[3] Lesk, A. M.: Introduction to protein architecture. New York: Oxford, 2001.

[4] Cystic Fibrosis Symptoms, prosinac 2010., http://www.cysticfibrosissymptoms.net/

[5] HTML5 – the canvas element, siječanj 2011., http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html

[6] Rawson, K.: *The google o3d bible*, prosinac 2010., gwt-o3d.googlecode.com/files/the_o3d_bible_by_kara_rawson.pdf

Distribuirani sustav za vizualizaciju
mikrobioloških podataka

Sažetak

Ovaj rad prikazuje primjenu računalne grafike u okviru bioinformatike. Rad se sastoji od sedam glavnih cjelina. U prve tri cjeline opisani su osnovni postupci metodologije, kao i najvažniji pojmovi iz područja bioinformatike. Četvrto poglavlje daje kratak opis osnovnih strukture i baza podataka koje se koriste u navedenom području te na koji način se oni vizualiziraju.

Poglavlje 5. opisuje projekt *Activ8* čiji je cilj izgraditi programsku podršku koja bi pomogla liječnicima, istraživačima, ali i oboljelima od cistične fibroze. Poglavlje 6. daje detaljan opis izrađene programske podrške temeljene na O3D i WebGL API-u. U njemu se također objašnjavaju principi koji su se koristili u postupcima optimizacije koda. Poglavlje 7. daje kratak pregled budućnosti *Activ8* projekta.

Ključne riječi: bioinformatika, WebGL, O3D.

Distributed system for visualisation of microbiological data

Abstract

This thesis describes the application of computer graphics in the area of bioinformatics. It is divided into seven main chapters. The first three chapters describe the basic principles as well as the main topics of bioinformatics. The fourth chapter gives a short introduction about data types and data bases used in this area as well as some methods on how they are being visualized.

Chapter five describes the *Activ8* project. The goal of this project is to build software which would help clinicians, researchers and cystic fibrosis patients. Chapter 6. provides a detailed overview of the created software based on WebGL and O3D API. It also describes the basic principles how the code was optimized. Chapter 7. gives a short preview about the future of the *Activ8* project.

Keywords: bioinformatics, WebGL, O3D.

PRIVITAK

Kazalo slika

Tabela 4. Mogućnosti pojedinih tehnologija razmatranih u okviru projekta Activ8.

API/Pogon	Vizualizacija i kvaliteta prikaza	Programiranje	Interaktivnost	3D modeliranje
O3D (Google)	Temelji se OpenGL ES specifikacijama. Kvaliteta prikaza ovisi o konkretnom hardveru računala. Ima mogućnost kreiranja programa za sjenčanje.	JavaScript, ali pomoću GWT-a se razvoj aplikacije može obaviti u Eclipsu (Java).	Pomoću JavaScripta mogu se prepoznati vanjski događaji (klikanje), te se stoga može implementirati navigacija i interakcija. Sadrži osnovnu klasu za selekciju objekata.	Može učitati konvertirane modele iz 3D Studio Max-a i Maye.
UNITY	Temelji se OpenGL specifikacijama. Kvaliteta prikaza ovisi o konkretnom hardveru računala. Ima mogućnost kreiranja programa za sjenčanje. Postoji podrška za fiziku, generiranje terena, vlastiti editor scena…	Ne može ostvariti prikaz u Internet pregledniku.	Sučelje za interakciju detaljno razvijeno.	Podržava formate poput : maya, 3ds, blender, Autodesk FBX, Cinema 4D...
Away3D (Flash)	Svi flash3D pogoni imaju lošu podršku za sjenčanje.	Mora se koristiti Flash.	Interakcija je moguća. Moguće iskorištavanje web kamere.	Podržava Collada, 3DS, Md2 i MQO formate.
Java 3D	Biblioteka iznad razine OpenGL-a. Sadrži graf scene. Ne podržava m*ulti-pass rendering.*	Jednostavan za razvoj aplikacije (Java, graf scene…)	Program se odvija u appletu, ne unutar Internet preglednika. Postoji već implementirana interaktivnost.	Podržava: 3DS, OBJ, VRML, X3D, NWN, FLT i lightwave 3d.
JOGL	Izgrađen iznad OpenGL-a. Postoji samo osnovna funkcionalnost.	Zahtjevno jer postoji samo osnovna funkcionalnost.	Interakcija pomoću miša i tipkovnice postoji. Selekcija objekata nije implementirana.	Ne postoji sučelje za učitavanje 3ds ili maya formata.
LWJGL	Temelji se OpenGL specifikacijama. Kvaliteta prikaza ovisi o konkretnom hardveru računala. Samo osnovna funkcionalnost.	Zahtjevno jer postoji samo osnovna funkcionalnost.	Interakcija pomoću miša i tipkovnice postoji. Selekcija objekata nije implementirana.	Može učitati konvertirane modele iz 3D Studio Max-a i Maye.
JME	Temelji se na JOGL biblioteci. Služi kao pogon za igranje. Nije moguće integrirati unutar Internet preglednika.	Razvoj u Javi, postoji velika količina već implementirane funkcionalnosti.	Interakcija pomoću miša i tipkovnice kao i selekcija objekata jest implementirana.	Podržava OGREXml, collada i MD5 formate. Postoje dodatci za Maya i 3D Max formate.
3D MaX	Nije dostupno.	Nije dostupno.	Nije dostupno.	Nema važnijih opaski.
Maya	Nije dostupno.	Nije dostupno.	Nije dostupno.	Nema važnijih opaski.