Izrada LISP modula za CAD aplikacije

Mrkonjić, Vedran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

IZRADA LISP MODULA ZA CAD APLIKACIJE

Diplomski rad

Vedran Mrkonjić

Osijek, 2018.

SADRŽAJ

1. UVOD

Ovaj rad se bavi problematikom korištenja LISP modula za *CAD* aplikacije. Rad se sastoji od teorijskog i praktičnog dijela. Teorijski dio opisan je u drugom i trećem poglavlju. Drugo poglavlje opisuje povijest razvoja *CAD* sustava, njegova implementacija i primjena u industriji te neke od njegovih značajki zbog kojih se koristi. U trećem poglavlju se opisuje LISP jezik. Prolazi se kroz njegovu povijest te se opisuju razni dijelovi AutoLISP-a koji ima važnu ulogu kod uporabe AutoCAD i progeCAD aplikacija.

U zadnjem poglavlju su predstavljeni primjeri LISP modula koji prikazuju kako pomoću AutoLISP-a odraditi željene zadatke u AutoCAD-u ili progeCAD-u. Kroz primjere se korisnik upoznaje s radom AutoLISP-a, AutoCAD-a i progeCAD-a.

2. CAD APLIKACIJE

Računalno potpomognuto oblikovanje ima važnu ulogu na području tehničkih znanosti i inženjerstva. Zbog velikih mogućnosti koje pruža ovaj način modeliranja, modeliranje postaje brže i jednostavnije. Ukoliko se pogriješi, greške se mogu lako ispraviti. Ovo poglavlje će prikazati razvoj i primjene računalno potpomognutog oblikovanja, te njegove mogućnosti.

2.1. Uvod u CAD

Računalno potpomognuto oblikovanje (eng. *CAD - Computer aided Design*) prema [1] se može definirati kao uporaba računalnog sustava za pomoć u izradi, modifikaciji, analizi i optimizaciji dizajna. *CAD* sustav se prema [2] sastoji od sklopovlja, specijalizirane programske podrške te vanjske jedinice koja je također u nekim slučajevima specijalizirana. Srž *CAD* sustava je programska podrška koja omogućuje grafički prikaz proizvoda, baze podataka za spremanje modela proizvoda i pokretanje vanjske jedinice za prikaz proizvoda. Dizajner je glavni sudionik u procesu, on sudjeluje u svim fazama procesa od identifikacije problema do implementacijske faze.
Primjeri primjene *CAD* sustava su: Analiza naprezanja komponenti, izračuni za prijenos topline i slično. Odabir odgovarajućeg *CAD* programa prema [2] će ovisiti čime se određena tvrtka/dizajner bavi.

Uloga *CAD*-a je da pruži dizajneru:

- Preciznu izradu i jednostavno oblikovanje grafičkog prikaza proizvoda. Tako korisnik može vidjeti gotovo stvarnog izgleda proizvoda, oblikovati ih, te prikazati dizajnerovu ideju na zaslonu bez izrade prototipa, posebice tijekom ranog stadija dizajniranja.
- Izvodi zahtjevne analize dizajna u kratkom vremenu.
- Spremanje i pozivanje informacija s dosljednošću i brzinom. Posebice kod korištenja Sustava za upravljanje podacima o proizvodu (engl. *PDM – Product Data Management*) koji mogu pohraniti cijeli dizajn i povijest obrade određenog proizvoda, za ponovno korištenje u budućnosti i nadogradnju.

2.2. Povijest CAD-a

Korijeni današnjeg *CAD*-a prema [3] sežu na početke civilizacije kada su inženjeri starog Egipta, Grčke i Rima komunicirali slikama. Neki od postojećih crteža u egipatskim grobnicama mogu se smatrati tehničkim crtežima. Dostupni radovi i zapisi Leonarda da Vincija prikazuju korištenje današnjih grafičkih konvencija poput izometričnog pogleda i kosog šrafiranja.

Ortografska projekcija koju danas koristimo je izmislio francuski matematičar Gaspard Monge (1746. – 1818.) koji je bio zaposlen kao dizajner od strane vlade. Ova metoda projekcije je postala dostupna za javnost početkom devetnaestog stoljeća nakon što ju je vojska skrivala trideset godina.

CAD je prošao kroz četiri velike faze razvoja. Prva faza je bila u pedesetim godinama prošlog stoljeća i može biti karakterizirana kao era početka interaktivne računalne grafike. Razvoj u prvoj polovici tog desetljeća je bio usporen zbog troškova i neadekvatnosti računala za interaktivno korištenje. *MIT* (engl. *Massachusetts Institute of Technology*) je bio u mogućnosti proizvesti jednostavne slike „*Whirlwind*“ računalom (računalo s vakuumskim cijevima iz ere Hladnog rata) 1950. godine pomoću sučelja sličnog televizorima s katodnim cijevima . U 1952. godini, na *MIT*-u je prikazan koncept numeričke kontrole na glodalici s tri osi. Pasivna grafika, prikazana na zaslonu s katodnim cijevima, je korištena sredinom pedesetih godina prošlog stoljeća u vojne svrhe. U drugoj polovici desetljeća je osvanuo koncept svjetlosnog pera.

Šezdesete godine predstavljaju najznačajnije razdoblje za istraživanje interaktivne računalne grafike. Činjenica da su računala izašla iz istraživačkih laboratorija je uvelike pridonjele razvoju u ovom desetljeću. Prekretnica istraživačkih postignuća se dogodila kada je Ivan Sutherland [4] razvio *Sketchpad* sustav, koji je bio objavljen 1962. godine u njegovom radu „*Sketchpad*: Grafički komunikacijski sustav između čovjeka i stroja“ [5] (engl. *Sketchpad: A Man-Machine Graphical Communication System*) .

Razvoj *Sketchpad* sustava je bio dramatičan događaj jer je pokazao da je moguća izrada crteža i promjena objekata interaktivno na zaslonu. Do sredine šezdesetih godina je bilo pokrenuto veliko istraživanje računalne grafike od strane raznih grupa. Tada se počeo pojavljivati i koristiti pojam „Računalno potpomognuto oblikovanje“ ili *CAD*.

Prvi sustavi su bili vrlo skupi, grafičko sučelje nije bilo napredno u to vrijeme i za korištenje sustava bilo je potrebno posebno sklopovlje i programska podrška koji su bili pruženi

većinom od prozvođača *CAD* aplikacija. Prvi *CAD* sustavi bili su namijenjeni za instalaciju na središnja računala (eng. *mainframe computer*), a pristupalo im se s terminala, dok se danas koristi tehnologija s mrežno povezanim samostalnim radnim stanicama (*UNIX* i *Windows* operacijski sustavi). AUTODESK je bio prvi proizvođač softvera koji je omogućio korištenje *CAD* sustava na osobnom računalu koji je nazvan AutoCAD (početkom 1980). Pojam „dizajna" je predstavljao više od osnovnog koncepta crteža.

Tijekom sedamdesetih godina, istraživanja na području računalne grafike su značajno napredovala te je viđen veliki potencijal na području interaktivne računalne grafike u poboljšanju proizvodnje. To desetljeće se može smatrati zlatnim dobom računalnog crtanja. „*Turnkey*" sustavi (računalni sustavi koji su unaprijed izrađeni i spremni za korištenje) su pružali crtačima i/ili dizajnerima trodimenzionalne centralizirane baze podataka prvenstveno u svrhu modeliranja i crtanja. Neki od ovih sustava su bili spori 16 bitni strojevi i većina je podržavala žičano modeliranje (modeliranje 3D objekta gdje je potrebno odrediti svaki ugao fizičkog objekta te gdje se dvije plohe međusobno spajaju) te ograničene mogućnosti plošnog modeliranja (modeliranje za prikaz čvrstih objekata kao npr. ilustracija te za arhtitektonsko modeliranje, manje je geometrijski ispravno od žičanog modeliranja te se koristi za prikaz izgleda objekta). Zbog ograničenja modeliranja, bile su dostupne samo aplikacije za izradu osnovnog dizajna. Takve aplikacije su većinom bile manualne, te nisu bile u mogućnosti nositi se sa stvarnim problemima industrijskog dizajna.

U kasnim sedamdesetima menadžment je u raznim industrijama počeo shvaćati utjecaj tada nove *CAD*/*CAM* (engl. *Computer-aided manufacturing* – računalno potpomognuta proizvodnja) tehnologije na povećanju produktivnosti. Od tada su inženjeri širili tehnologiju na više od samog crtanja. Zahtjevali su različite aplikacije za dizajn i proizvodnju od dobavljača koji su uspješno obavljali posao u postojećim granicama sklopovlja i programske podrške.

U osamdesetim godinama je glavni cilj bio integrirati i/ili automatizirati različite elemente dizajna i proizvodnje za daljnji razvoj industrije. Glavni fokus istraživanja je bilo proširenje *CAD/CAM* sustava izvan trodimenzionalnog geometrijskog dizajna i pružiti više inženjerskih primjena.

Osnovni potencijal čvrstog modeliranja leži u činjenici da pruža jedinstven i nedvosmislen geometrijski prikaz čvrstog tijela, koji pomaže automatizirati i/ili podržati dizajn i primjene u proizvodnji. Sklopovlje je išlo u korak s razvojem programske podrške i razvojem aplikacija.

2.3. Ciljevi CAD-a

Prema [2] izvorno je tehnika bila usmjerena na automatizaciju brojnih zadataka koje je dizajner obavljao, posebice modeliranje proizvoda. Danas *CAD* sustavi pokrivaju većinu aktivnosti u ciklusu dizajna, oni bilježe sve podatke o proizvodu, i koriste se kao platforma za suradnju između udaljenih projektnih timova. Većina njegovih primjena je u proizvodnji i uobičajeno ime aplikacije je *CAD/CAM*. Većina funkcija se ne ostvaruje pomoću samo jednog sustava već je vrlo često da tvrtka koristi više od jednog sustava, posebice kada imamo *CAD* i *CAE* aplikacije (engl. *Computer-aided Engineering* – računalno potpomognuto inženjerstvo).

CAD sustavi mogu skratiti vrijeme oblikovanja proizvoda. Tako se proizvod može ranije predstaviti tržištu, pružajući mnogo prednosti za tvrtku. Što prije proizvod izađe na tržište duže će imati koristan životni vijek, ukoliko je kvaliteta zadovoljavajuća.

Kao što je prethodno spomenuto, prve primjene *CAD*-a su bili 2D nacrti, dok danas su većina 3D čvrsti i parametarski prikaz stvarnog dijela. Kompletni sklopovi mogu biti modelirani i može se provesti potpuna analiza virtualnog prototipa. 3D prikaz se može izvest na druge platforme i služiti kao komunkacija između grupa ljudi iz različitih odjela tvrtke.

CAD sustavi omogućuju primjenu istovremenog inženjeringa i mogu imati značajan utjecan na konačnu cijenu proizvoda, funkcionalnost i kvalitetu.

2.4. Kreiranje baze podataka za proizvodnju

Bitni razlog za korištenje *CAD* sustava prema [1] je to da on pruža mogućnost razvoja baze podataka potrebne za proizvodnju proizvoda. U konvencionalnom ciklusu proizvodnje, inženjerski nacrti su pripremljeni od strane dizajnera koje zatim koriste inženjeri proizvodnje za razvoj procesnih planova. Aktivnosti uključene u dizajniranje su bile odvojene od aktivnosti vezanih uz planiranje procesa. Što znači da se koristio postupak u dva koraka. Ovo je ujedno zahtjevalo mnogo vremena te dupliciranje od strane inženjera za proizvodnju i dizajn. U integriranom *CAD/CAM* sustavu, uspostavljena je izravna veza između dizajna

proizvoda i proizvodnje. Cilj *CAD/CAM* sustava je ne samo da automatizira određene faze dizajna i proizvodnje nego da također automatizira prijelaz od dizajna do proizvodnje.

2.5. Sustavi vezani za ciklus razvoja proizvoda

Raspon sustava vezanih za ciklus razvoja proizvoda je prilično opsežan.
Uobičajena podjela je prema [2] na: razvoj koncepta ili proizvoda industrijskog dizajna, *CAD* sustave, računalno potpomognute inženjerske alate, analizu konačnih elemenata, alate i strojeve za brzu izradu prototipa, primjenu *CA*D-a i postupak implementacije. Svaki od navedenih postupaka biti će opisan u narednim podnaslovima.

2.5.1. Razvoj koncepta ili proizvoda industrijskog dizajna

Ovo su većinom sustavi plošnog modeliranja za mehaničke proizvode, s vrlo dobrim mogućnostima renderiranja. Proces inače započinje s grubom skicom proizvoda, zatim se primjenjuju boja i tekstura, te se kreira 3D model iz 2D skice na kojemu se koristi napredno foto-realistično renderiranje i animacija za daljnje vrednovanje, prezentaciju i prodaju koncepta.

Fotografija 2.1. Prikaz skice proizvoda, dodavanja teksture i boje, te renderiranje 3D modela.

a) b) c)

Izvor: Dr. Nicos Bilalis, Computer Aided Design, Technical University of Crete, 2000.

Računalno potpomognuti industrijski sustavi za oblikovanje su primjenjeni kod mnogo industrijskih proizvoda, od svakodnevnih potrošačkih proizvoda, sportske robe, računala, opreme, i složenih dijelova poput automobila.

To su inače integrirani proizvodi koji imaju mogućnost prikazati cijeli proces od početnog koncepta do izrade objekta, a kombiniraju:

- digitalno crtanje, omogućujući eksperimentiranje koje nije moguće kod tradicionalnih alata
- slobodno plošno modeliranje s vrlo fleksibilnim alatima za modeliranje
- jedinstven realizam u prikazu dizajna, vrednovanju, recenziji i odobrenju
- kvalitetu, točnost i preciznost koji su potrebni za integraciju s inženjerskim i proizvodnim procesima
- alati za obrnuti inženjering, koji pretvaraju podatke iz digitalizacije u 3D digitalni model
- prijenos podataka u *CAD* sustave

2.5.2. CAD sustavi

Trenutni sustavi, posebice za mehaničke proizvode su 3D sustavi i oni šire svoju dominaciju na druge sektore. 3D modeliranje može biti žičano modeliranje, plošno modeliranje ili modeliranje čvrstim tijelima. Većina *CAD* sustava srednjeg ranga su parametarski sustavi i sustavi za čvrsto modeliranje pomoću značajki.

Žičano modeliranje je bio prvi pokušaj za prikaz 3D objekta. Prikaz je bio neodgovarajuć s mnogo nedostataka poput preciznosti, podešenosti prikaza, i sličnog. Jednostavnim riječima 2D-žičani model je napravljen oblikovanjem kostura tijela, koji se sastoji samo od rubova. Ova tehnika je sada jedan od koraka pri izradi plošnog modela ili čvrstog modela.

Plošnim modeliranjem se modelira omotač dijela. Današnji sustavi većinom koriste *NURBS* (matematički model koji se često koristi u računalnoj grafici, te pruža veliku preciznost i fleksibilnost pri radu [6]), koji su u mogućnosti modelirati skoro svaki industrijski dio, poput površine zrakoplova i površine automobila (karakterizirane kao površine A klase [7]), izgradnje brodova, plastičnih dijelova i pakiranja općenito, metalnih dijelova, cipela, i slično. Sustavi koji koriste *NURBS* model su najsposobnija vrsta sustava za prikaz industrijskih dijelova. Njegovo korištenje nije jednostavno i zahtjeva značajno znanje *NURBS* matematike. Oni omogućuju kreiranje površina, koje trenutno nisu dostupne kod sustava čvrstog modeliranja. Uređivanje modela vrši se oblikovanjem krivulja određivanja, tako da se mijenja numerička vrijednost parametara pomoću grafičkih ili matematičkih zakona koji

kontroliraju stvorene oblike. Sustavi također uključuju jednostavne alate za procjenu oblika, veličine i zakrivljenosti složenih modela. Površine kreirane pomoću modula za slobodno plošno modeliranje mogu biti integrirane u čvrsti model. Ovi modeli nisu u mogućnosti modelirati umjetničke dijelove (poput nakita), ili organske oblike poput akcijskih figurica, ljudskog tijela i lica.

Smatra se da sustavi za čvrsto modeliranje pružaju najcjelovitiji prikaz objekta. Oni kombiniraju modeliranje i topologiju. Rani sustavi su bili temeljeni na primitivima za prikaz prostora. Trenutni sustavi su granične zastupljenosti (B-Rep) tipa (čvrsto tijelo je prikazano kao skup povezanih plošnih elemenata). Tijekom devedestih godina svi su sustavi za čvrsto modeliranju karakterizirani kao parametarski i sustavi za modeliranje pomoću značajki.

S parametarskom tehnologijom korisnik dodjeljuje parametre za određivanje dimenzija, odnose između parametara i odnose između dijelova (u smislu pozicije i veličine). Tako se može definirati novi dio dodjeljivanjem novih vrijednosti parametrima ili definiranjem cijele porodice dijelova preko tablice dimenzija. S modeliranjem pomoću značajki korisnik ima pristup višoj razini izražavanja pri modeliranju. Ove značajke imaju određen broj svojstava uključujući oblik, dimenzije i poziciju.

Većina sustava ima integrirane module za dizajn lima [8](engl. *sheet metal design*), omogućujući dizajneru da definira i simulira proizvodne sekvence, preoblikuje modele te generira točne podatke uzorka za daljnju primjenu.

Dijelovi kreirani u sustavima čvrstog modeliranja mogu biti izvezeni u sustave nacrta za izradu crteža. Ovaj modul kreira dimenzije koje su povezane s geometrijskim modelom, osiguravajući da su ažurirani na promjenu modela i smanjujući vrijeme potrebno za ažuriranje crteža.

2.5.3. Računalno potpomognuti inženjerski alati

Inženjerska analiza se bavi analizom i vrednovanjem dizajna inženjerskog proizvoda. Za ovu svrhu, više računalno baziranih tehnika je korišteno za izračun operacijskih, funkcionalnih i proizvodnih parametara proizvoda. Analiza konačnih elemenata je jedna od najčešće korištenih tehnika inženjerske analize. Uz analizu konačnih elemenata još se koriste i analiza tolerancije, optimizacija dizajna, analiza mehanizma koje su neke od računalno potpomognutih tehnika dostupnih inženjerima u svrhu analize i vrednovanja dizajna inženjerskih proizvoda.

2.5.4. Analiza konačnih elemenata

Analiza konačnih elemenata je snažan proces numeričke analize koja je u širokoj uporabi za inženjerske primjene. Analiza konačnih elemenata se koristi za analizu i proučavanje funkcionalnih performansi objekta tako što se objekt dijeli na manje gradivne blokove, koji se nazivaju konačni elementi. Jezgra analize konačnih elemenata je idealizacija objekta ili kontinuuma konačnim brojem diskretnih varijabli. Za tu svrhu, objekt je prvo podijeljen na mrežu elemenata koji tvore model stvarnog objekta. Svaki element je jednostavnog oblika poput kvadrata, trokuta, kocke ili nekog drugog standardnog oblika za koji program konačnih elemenata ima informaciju za pisanje jednadžbe u obliku matrice krutosti.

2.5.5. Alati i strojevi za brzu izradu prototipa

Brzi prototip omogućuje jednostavan „ispis“ trodimenzionalnog modela poput ispisa na papiru. Brz je i isplativ način za poboljšanje načina na koji dizajner prikazuje svoje ideje, unutar i izvan organizacije. Ono revolucionira proces razvoja, pomažući dizajnerskom timu da iskoristi više mogućnosti. Također pridonosi boljem razumijevanju proizvoda i time se omogućuje brže odobrenje ideja proizvoda.

2.6. Primjena CAD-a

CAD se primjenjuje u mnogim industrijskim sektorima.

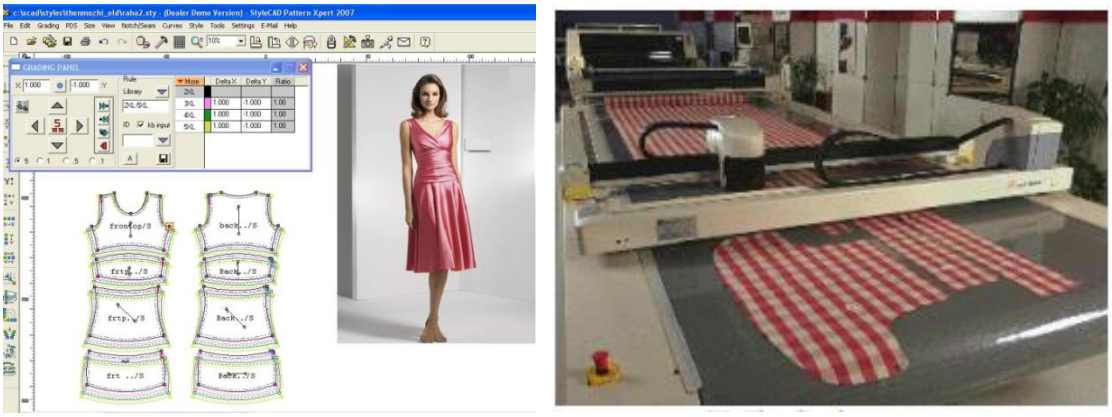
- Strojarski sektor je najveći korisnik *CAD* sustava. Aplikacije su obično povezane s proizvodnjom, kreirajući *CAD/CAM* sustave. Aplikacije pokrivaju sve tipove proizvodnih operacija, poput glodanja, okretanja, bušenja i sličnog. Korisnik može testirati dio programa na zaslonu prije nego se prebaci na stroj i spriječiti sudar, rušenje i slično. Većina sustava imaju integrirani *post* procesor za komunikaciju s alatnim strojevima.
- Arhitektura i građevina su drugo najveće područje primjene *CAD* sustava. Primjene variraju od dizajna jedne jednostavne građevine, do velikih projekata, uređenja prostora, statičke i dinamičke analize itd.
- Elektroničko inženjerstvo je treće najveće područje primjene. Računalo obavlja sve radnje potrebne za dizajniranje integriranih krugova. Složenost dizajna nameće

korištenje *CAD* sustava. Trenutni sustavi također uključuju više dizajna kako bi pružili pomoć korisnicima za njihove nove dizajne.

- Tekstilna industrija je također veliki korisnik. Broj dostupnih sustava je ograničen. Sustavi za odjeću su vrlo skupi, jer koriste specijaliziranu opremu, poput velikih plotera, rezača za krojeve, i automatski strojevi za rezanje tkanine. Raspon njihove primjene je od modnog dizajna do proizvodnje. Najveća uporaba sustava je za dizajniranje uzoraka i planiranje položaja, gdje se ostvaruje najviše ušteda. Apsolutno je neophodno da danas sve tvrtke koje podugovaraju veće dobavljače mogu obraditi elektronske podatke (dizajn uzoraka). Planiranje položaja može uštediti na materijalu i može opravdati investiranje u *CAD*. Danas primjena sustava za modni dizajn postaje vrlo popularna. Ovi sustavi mogu proizvesti cijelu kolekciju na papiru, time štedeći mnogo novca na izradu probnih uzoraka i vrlo često su povezani sa specijaliziranim tintnim pisačem koji ima mogućnost ispisa na tkaninu za brzu izradu uzorka. Slično *CNC* (engl. *Computer numerical control* – upravljanje mehaničkim alatima pomoću računala koja izvršavaju unaprijed programirane sekvence naredbi) strojevima koji se koriste u proizvodnji metalnih elemenata, računalom upravljani strojevi postoje i u tekstilnoj industriji. Ovi strojevi obavljaju automatski postavljanje tkanine i rezanje te su upravljani direktno iz *CAD* sustava.

Fotografija 2.2.: Prikaz izrade uzorka u CAD-u, te rezanje uzorka

a) Izrada uzorka b) Rezanje uzorka



Izvor: Dr. Nicos Bilalis, Computer Aided Design, Technical University of Crete, 2000.

Troškovi implementacije sastoje se od nabave programske podrške, opreme i obuke. Nijedan od ovih troškova se ne bi trebao podcijeniti. Cijene sustava neprestano padaju, ali cijena obuke raste. Troškovi su veći za specijalizirane sustave poput programa za analizu. Što se tiče sustava podrške, troškovi podrške i savjetovanja su vrlo veliki. Proces implementacije je dug i treba mnogo vremena da se sustav postavi i koristi učinkovito.

2.7. Postupak implementacije

Uspješno uvođenje cjelokupnog *CAD/CAM* sustava u industrijski proces je dug postupak. On započinje, kao i svaka IT aplikacija, s vrednovanjem potreba, tehničkim zahtjevima sustava, odabirom najpogodnijeg sustava i implementacijom.

Tijekom vrednovanja potreba sustava, implementacija cijelog sustava treba biti podijeljena na korake. Određeni koraci se moraju pratiti prije implementacije modula. Postojeći IT sustavi i oprema, koji zahtjevaju razmjenu podataka s *CAD* sustavom moraju biti uzeti u obzir pri sastavljanju zahtjeva, kao i trenutni trendovi u razvoju *CAD* sustava .

Tehnički zahtjevi sustava bi trebali pokriti većinu potreba korisnika. U pojedinim primjenama oni bi trebali biti neovisni od dobavljača. U zemljama u razvoju gdje je broj korisnika sofisticiranih sustava ograničen, lokalna podrška je vrlo važna.

Vrednovanje alternativnih rješenja i odabir najprikladnijeg sustava ne bi trebao biti temeljen samo na vrednovanju dokumenata. Mora se napraviti sustavno vrednovanje tri do četiri sustava, temeljene na jednom ili dva dijela, predstavnika spektra proizvoda koje tvrtka proizvodi. Vrednovanje dokumenata različitih sustava nije lako za obaviti, ukoliko tvrtka nije korisnik *CAD*-a, nema vrlo dobro poznavanje *CAD* sustava te ne prati razvoj unutar *CAD* sustava.

Ponekad vrsta posla kojom se tvrtka bavi upućuje na odabir *CAD* sustava. Veliki proizvođaći s mnogo podugovaratelja predlažu tvrtkama da nabave iste *CAD* sustave kao što i oni koriste. Oni ciljaju na izbjegavanje problema kod prijenosa podataka iz jednog *CAD* sustava u drugi.

Implementacija je najteža faza. Problemi koji se moraju riješiti uključuju, obuku, organizaciju *CAD* tima, upravljanje sustavom i procedure koje moraju postojati.

Obuka

Obuka je najvažniji korak za uspješnu implementaciju *CAD* sustava. Za nove instalacije, odabir osoblja i uspješnost obuke su kritični jer će zapreke možda biti vrlo teško nadvladati. Naglasak bi trebao biti stavljen na usavršavanje stručnog znanja i povećanje samopouzdanja umjesto ciljeva proizvodnje koje će biti teško postići.

Nakon početnog postavljanja sustava i obuke, produktivnost bi ubrzo trebala dostići razinu proizvodnje prije nego je uveden *CAD*. Ali, mnogo puta zapreke se javljaju zbog gubitka podataka, grešaka operatera ili manjak stručnosti koji će uništiti proizvodnju te isfrustrirati menadžment i korisnike. Razlozi ovih zapreka moraju biti pronađeni i ispravljeni. Svaki problem koji se nađe trebao bi se koristiti kao prilika za učenje.

Operateri imaju sklonost da koriste poznate naredbe umjesto da probaju naučiti nove i brže načine za obavljanje njihovog posla. Zadatak će obično pasti na manadžera da prepozna operacije koje mogu biti ostvarene produktivnije te da istraži i razvije nove metode za korištenje.

Organizacija

Ako je radna jedinica pravilno organizirana, menadžer će moći posvetiti više vremena na proizvodnju i tehničke probleme. Kooperacija između odjela je važna za odnose između *CAD/CAM*-a unutar tvrtke. Očekujući buduće primjene *CAD/CAM* opreme, grupa će biti bolje pripremljena za daljnji rast na organizirani način. Budući planovi za proširenje bi trebali biti uzeti u obzir prije znatnih ulaganja u izgled ili dizajn postrojenja. Problemi s neodgovarajućim računalnim sustavima mogu biti izbjegnuti s odgovarajućim planiranjem i komunikacijom između odjela koji će koristiti *CAD/CAM* bazu podataka.

Operacije

Uspjeh *CAD*-a nije automatski. Tvrtka mora naučiti koristiti tečno dijelove biblioteke, izbornike, i osnovne naredbe sustava prije nego pokuša programirati sustav. U slučaju da korisnik odabire projekte za automatizaciju, on mora biti siguran da je to dovoljno posla da nadoknadi za trošak obuke i troškove razvoja programske podrške, posebice ukoliko je potrebno opsežno programiranje.

Preporučeno je da se prvo automatiziraju zadaci koji će uštediti najveću količinu novca i koji će imati najveću šansu uspješnog izvršenja na sustavu. Treba imati na umu da posao neće biti obavljen ako su zadaci iznad mogućnosti operatera.

Rasporedi moraju sadržavati vrijeme za razvoj sustava ili će se mogućnosti brzo smanjiti. Svi članovi *CAD* tima bi trebali sudjelovati u analiziranju kako bi pronašli „bolji način da ga naprave“ na *CAD* sustavu.

Procedure

Napisane procedure su važne jer one pružaju jasne, neosporne upute za *CAD* osoblje i *CAD* korisnici znaju što se očekuje od njih. Ako korisnici postanu zbunjeni što se tiče zahjeva zadatka, oni mogu iznjeti njihove primjedbe jasnije upućivanjem na dijelove procedure koji trebaju uljepšavanje ili reviziju. Procedure bi trebale biti ažurirane prema potrebi za pojednostavljenje operacija i pružiti najbolje uklapanje s drugim odjelima koji koriste *CAD* bazu podataka.

Procedure su također korisne jer kada su jednom dovršene mogu se ponovno koristiti. One pomažu obučiti nove korisnike i osvježiti znanje povremenih korisnika, i informacija nije izgubljena kada važni ljudi napuste tvrtku. Dobre procedure će pružiti zdravu osnovu za produktivan *CAD* razvoj unutar tvrtke.

2.8. Tehničko crtanje na računalu

U ovom poglavlju prema [9] je riječ o načinu predstavljanja crteža i slika u računalu, grafičkih jedinica te planiranja izrade crteža.

2.8.1. Predstavljanje crteža na računalu

Postoje dva osnovna načina za predstavljanje crteža u računalu, a to su vektorski i rasterski. Pri tome treba razlikovati način predstavljanja crteža u računalu od načina predstavljanja crteža na izlaznoj jedinici. To je najčešće slučaj kod osobnih računala gdje se iz programa koji u računalu predstavljaju sliku vektorski, dobije slika na rasterskoj izlaznoj jedinici (zaslon ili uređaj za iscrtavanje na papir).

Kod programa koji predstavljaju slike vektorski, pamte se linije od kojih je slika sastavljena i njihovi atributi (debljina, vrsta linija, boja), a kod zatvorenih kontura i boja unutrašnjosti konture. Na primjer, ako je na slici samo kružnica, potrebno je zapamtiti: tip objekta, koordinate njenog centra, polumjer i boju kojom je popunjena. Jasno je da količina podataka koju treba zapamtiti ovisi od složenosti slike.

Također grafički objekti se modificiraju jednostavno promjenom odgovarajućeg parametra u zapisu objekta. Ako se želi uvećati ili umanjiti objekt, računalo će jednostavno izračunati nove vrijednosti parametara i zatim nacrtati sliku u skladu s novim vrijednostima. Pri tome ni jedan dio slike i ništa od informacija o slici nije oštećeno ili nepovratno izgubljeno. Prilikom iscrtavanja dokumenta na papir, on će uvijek biti nacrtan najvišom mogućom rezolucijom koju uređaj za ispisivanje dozvoljava, jer se izračunate točke crteža zamjenjuju približnim vrijednostima točaka (pikselima) tek prilikom pripreme dokumenta za izlazak na zadani grafički uređaj.

2.8.2. Grafičke jedinice

Za vizualno predstavljanje računalnih crteža danas na tržištu postoji veliki broj kvalitetnih uređaja koji rade na temelju različitih tehnologija. To mogu biti uređaji za predstavljanje crteža na zaslonu ili njihovo iscrtavanje na papiru.

Grafički izlazni uređaji koji se koriste u svakodnevnom radu zasnovani su na rasterskoj tehnologiji. To su uobičajeni monitori koji se koriste za rad s osobnim računalima, sve vrste pisača kao i određeni tipovi plotera.

Jedna od bitnih karakteristika je i rezolucija. Kod vektorskih uređaja ona je određena najmanjim rastojanjem na kojemu se dvije točke mogu prikazati kao različite. Točke koje su na manjoj udaljenosti se prikazuju kao jedna točka. Rezolucija je veća što je udaljenost manja i izražava se u dijelovima inča ili centimetrima. Kod rasterskih uređaja rezolucija je određena brojem piksela. Što je veći broj piksela, za istu veličinu zaslona, veća je i rezolucija, a time je i kvaliteta slike bolja.

2.8.3. Planiranje izrade crteža

Prije početka izrade svakog crteža, makar i najjednostavnijeg, treba pažljivo projektirati postupak crtanja. Projektiranje crteža obuhvaća:

- određivanje veličine crteža
- određivanje slojeva, njihovih atributa i objekata koji se na njima nalaze
- uočavanje simetrija na crtežu
- uočavanje objekata koji se ponavljaju
- uočavanje objekata koji se dobivaju rotacijom drugih objekata
- uočavanje objekata koji se dobivaju transformacijama drugih objekata
- uočavanje objekata koji se dobivaju manjim izmjenama drugih objekata

Slojevi (engl. *Layers*)

Vrlo često se crtež sastoji od elemenata koji se izdvajaju od ostalih po svojim funkcionalnim ili drugim karakteristikama. Takvi elementi mogu se izdvojiti u posebnu grupu. Izdvajanjem se dobiva mogućnost da se pri crtanju na sve grupirane elemente utječe istovremeno, jednom naredbom. Time je funkcionalna povezanost koja postoji u stvarnosti, ostvarena i na crtežu. Ove grupe elemenata nazivaju se slojevi.

Simetrija na crtežu

Važno je uočiti postoji li simetrija na crtežu koji treba nacrtati. Simetrija može biti centralna ili osna. Ako se radi o osnoj simetriji, mogu postojati jedna,dvije ili više osi simetrije. U ovakvim slučajevima treba nacrtati samo jedan od karakterističnih dijelova, a zatim koristeći tehniku zrcaljenja, kompletirati crtež.

Ponavljanje istih ili sličnih objekata

U ovom slučaju crtanje se svodi na formiranje jednog elementa i njegovo preslikavanje na željene pozicije. Pri tome, element se može preslikati tako da bude identičan originalu, ali i da promijeni neke svoje karakteristike. Preslikani element se može uvećati, umanjiti, izdužiti ili sažeti.

Kada se radi o objektima koji su slični u osnovi, ali se razlikuju u detaljima, treba procijeniti isplati li se više prekopirati objekt i na njemu izvršiti potrebne izmjene, ili su izmjene tolike da je bolje nacrtati cijeli objekt iz početka.

2.9. progeCAD

progeCAD je prema [10] najpovoljniji klon AutoCAD-a. Trenutno ima preko 150.000 korisnika. progeCAD omogućuje jednake funkcionalnosti kao i AutoCAD, ali po nižoj cijeni. Korisnici koji su već koristili AutoCAD se tako mogu lagano prilagoditi ovom programu bez da uče novi program. Program razvija talijanska tvrtka progeCAD srl. progeCAD podržava rad s AutoCAD DWG formatom, tako se crteži rađeni u AutoCAD-u mogu razmjenjivati između AutoCAD-a i progeCAD-a. progeCAD također koristi AutoLISP.

2.9.1. Razlika između AutoCAD-a i progeCAD-a

Najveća razlika između AutoCAD-a i progeCAD-a je cijena koja je 15 puta veća za AutoCAD. AutoCAD je nešto brži u 3D načinu rada od progeCAD-a, dok su za 2D način rada razlike minimalne.

Dodatne funkcije koje progeCAD ima su :

- konverzija PDF u DWG i obrnut, te printanje DWG u JPG
- plot i print u progeCAD *Professional*
- *Express* alatna traka i *Add on* alatna traka
- Alat za upravljanje blokovima iCADlib (sadrži više od 20,000 blokova)
- *Traceparts* za progeCAD – preko iCADlib alaza omogućuje pristupanje internet bazi blokova
- *EasyArch* – Automatizirani alat namijenjen arhitektima i građevinskim inženjerima, napravljen da poveća produktivnost u svakodnevnom radu
- *Image management*
- Integracija *Google Eartha*

Tablica 3.1.: Razlika između terminologije progeCAD-a i AutoCAD-a

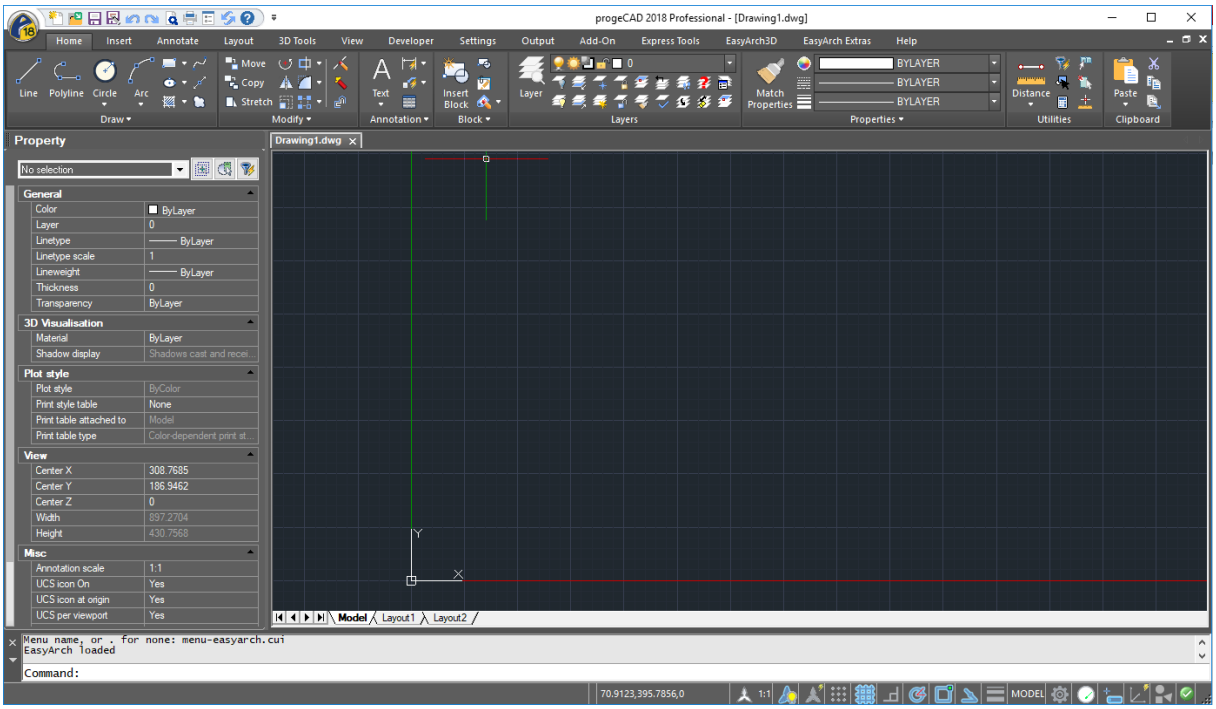
progeCAD pojam	**AutoCAD ekvivalentan pojam**
command bar	command line
entity	object
entity snap	object snap
fixed attribute	constant attribute
freehand	sketch
infinite line	XLine
orthogonal	ortho
parallel	offset
plane	solid (2D)
point snap	node snap
prompt box	context menu
quick selection	quick select

2.9.2. Najvažnije sličnosti AutoCAD-a i progeCAD-a

Najvažnije sličnosti između ova dva programa su:

- DWG, DXF i DWF formati datoteka
- Korisničko sučelje
- Naredbe i sistemske varijable
- Način printanja
- Alatne trake i ikone u alatnim trakama
- Izbornici, fontovi, tipovi linija i hatch-a
- LISP i VBA programiranje

Fotografija 2.2.: Prikaz korisničkog sučelja progeCAD-a



3. LISP

Lisp (engl. *List Processing Language* – jezik za obradu listi) je jezik velikih mogućnosti. Jedan je od najstarijih računalnih jezika, a koristi se i danas. Temeljen je na matematičkoj teoriji računanja. Kroz povijest se razvijao u razne dijalekte, a dok će neki dijalekti biti spomenuti, detaljnije će biti obrađen AutoLISP. AutoLISP ima važnu ulogu u AutoCAD-u jer uvelike proširuje same mogućnosti aplikacije.

3.1. Povijest LISP jezika

Lisp prema [11] je porodica jezika s dugom povijesti. Rane ideje za Lisp je razvio John McCarthy [12] tijekom Dartmouth ljetnog istraživačkog projekta 1956. godine na temu umjetne inteligencije. McCarthyjeva motivacija je bila razviti algebarski jezik za obradu lista za rad s umjetnom inteligencijom. Implementacija ranih dijalekta se provodila na IBM 704, IBM 7090. Glavni Lisp dijalekt između 1960. i 1965. godine je bio Lisp 1.5. Do ranih sedamdesetih godina bila su dva prevladavajuća dijalekta Lispa: MacLisp i Interlisp.

MacLisp se poboljšao na ideji Lispa 1.5 o specijalnim varijablama i rukovanjem pogreškama. Također uveden je koncept funkcije koja može primiti promjenjiv broj argumenata, makronaredbu, nizove, brzu aritmetiku, i naglasak na brzinu izvršenja.

Interlisp je uveo mnogo ideja u Lisp razvojna okruženja i metodologiju. Jedna od ideja Interlispa koja je utjecala na *Common Lisp* je provedba iteracija koju je implementirao Warren Teitelman koja je inspirirala petlju, makro koji se koristi i na Lisp strojevima i u MacLispu, i sada u *Common Lisp*-u.

PDP-10 računala i njihov prethodnik PDP-6 su, prema dizajnu, dobro odgovarali Lispu jer su imali 36 bitne riječi i 18 bitne adrese. Ova arhitektura je omogućila spremanje kontra ćelija u jednu riječ. PDP-6 i PDP-10 su imale brze, moćne instrukcije stoga koje su omogućavala brzo pozivanje funkcije. Ali ograničenost PDP-10 je očita 1973. godine, podržavao je mali broj istraživača koji su koristili Lisp, i mali 18 bitni adresni prostor je ograničavao veličinu programa. Jedan odgovor na problem adresnog prostora bio je *Lisp Machine*, računalo za posebne svrhe dizajnirano da pokreće Lisp programe. Drugi odgovori su bili računala opće namjene s adresnim prostorima većim od 18 bita.

Koncept *Lisp Machine* je razvijen kasnih šezdesetih godina. U ranim sedamdesetim godinama, Peter Deutsch [13], zajedno s Danielom Bobrowom[14], implementirao je Lisp na Alto, miniračunala za jednog korisnika, koristeći mikrokod za interpretaciju *byte*-kod

implementacijskog jezika. Ubrzo nakon toga, Richard Greenblatt[15] je započeo rad na drukčijem sklopovlju i dizajnu skupa instrukcija na *MIT*-u. Iako Alto nije bio pun pogodak kao *Lisp Machine*, dijalekt Interlispa poznat kao Interlisp-D je postao dostupan na strojevima D-serije proizvedenih od strane Xeroxa[16].

Tijekom sedamdesetih godina *Lisp Machine Lisp* (proširenje MacLispa) se počeo širiti prema mnogo potpunijem jeziku. Sofisticirane lambda liste, setf, višestruke vrijednosti, i strukture poput onih u *Common Lisp* su rezultat ranih eksperimentiranja *Lisp Machine* grupe sa stilovima programiranja. Oko 1980. godine, Scott Fahlman[17] i drugi na CMU (engl. Carnegie Mellon University) su počeli raditi na pokretanju Lisp-a na SPICE (engl. Scientific Personal Integrated Computing Environment) radnoj postaji. Jedan od ciljeva projekta je bio dizajnirati jednostavniji dijalekt od *Lisp Machine Lisp*.

Grupa Macsyma na *MIT*-u je započela projekt tijekom kasnih sedamdesetih godina zvan „Nova implementacija Lispa (NIL)“ za VAX [18]. Jedan od navedenih ciljeva NIL projekta je bila popraviti mnogo starih grešaka s Lisp-om dok zadržava važnu kompatibilnost s MacLisp-om. U isto vrijeme, istraživačka grupa na Sveučilištu Stanford i *Lawrence Livermore National Laboratory* pod vodstvom Richarda P. Gabriela[19] je započela dizajniranje Lisp-a s ciljem pokretanja na S-1 Mark IIA superračunalu[20]. S-1 Lisp, iako nikada u potpunosti funkcionalan, bio je podloga za testiranje prilagodbi naprednih tehnika prevođenja za Lisp implementacije. U konačnici S-1 i NIL grupa su surađivale.

Prvi napori prema standardizaciji Lispa su napravljeni 1969. godine, kada su Anthony Hearn[21] i Martin Griss[22] na Sveučilišitu Utah definirali *Standard Lisp*. Tijekom sedamdesetih godina Utah grupa je implementirala prvi prevoditelj za *Standard Lisp*, a zatim proširenu implementaciju poznatu kao *Portable Standard Lisp* (*PSL*). Do sredine osamdesetih, *PSL* se mogao pokrenuti na desetke različitih računala.

PSL i Franz Lisp (dijalekt kao MacLisp za Unix strojeve) su bili prvi primjeri široko dostupnih Lisp dijalekata na više platformi.

Jedan od navažniji događanja u razvoju Lispa se dogodio u drugoj polovici sedamdesetih godina i zvao se *Scheme*. *Scheme* je jednostavni dijalekt Lisp-a čiji je dizajn donio Lispu neke nove ideje semantika programskog jezika iz šezdesetih godina. Gerald J. Sussman[23] je bio jedan od glavnih inovatora iza mnogo drugih napredaka Lisp tehnologije. Veliki doprinos dijalekta *Scheme* je bilo leksičko ograničenje, leksičko zatvaranje, i pojednostavljena sintaksa. Neki od ovih doprinosa su imali veliki utjecaj na dizajn *Common Lispa*.

U kasnim sedamdesetim godinama koncepti objektno-orijentiranog programiranja su počeli imati snažan utjecaj na Lisp. Na *MIT*-u, određene ideje iz *Smalltalka*[24] su našle svoj put u

nekoliko programskih sustava u širokoj uporabi. *Flavors*, objektno-orijentirani programski sustav s višestrukim nasljeđivanjem, koje je razvijeno na *MIT*-u za *Lisp machine* zajednicu. U Xeroxu, iskustvo sa *Smalltalkom* i *Knowledge Representation Languagem* (KRL) je vodilo prema razvoju Lisp objektno-orijentiranog programskog sustava (LOOPS) i kasnije Common LOOPS.
Ovi sustavi su utjecali na dizajn *Common Lisp Object System* (CLOS). CLOS je razvijen posebno za X3J13[25] normiranje, i bio je odvojeno zapisan u „*Common Lisp Object System Specification*“. Međutim, manji detalji njegovog dizajna su malo promjenjeni od tog izdanja, i taj zapis se ne bi trebao koristiti kao autoritativna referenca za semantiku CLOS-a.

Tijekom osamdesetih godina došlo je do naglog razvoja umjetne inteligencije, Lisp je ostao omiljeni alat za programere koji pišu programsku podršku za rješavanje teških problema poput automatiziranog dokazivanja teorema, planiranje i računalni vid. Ovi problemi su zahtjevali mnogo programske podrške koja je teška za napisati; za napredak, programeri umjetne inteligencije trebali su snažan jezik, i tako su razvijali Lisp u jezik koji im je bio potreban. I Hladni rat je pomogao – tako što je Pentagon financirao DARPA-u (engl. *Defense Advanced Research Projects Agency*), velik dio tog novca je išao ljudima koji su riješavali probleme poput silmulacija bojnog polja velikih razmjera, automatsko planiranje, i prirodna jezična sučelja. Ovi ljudi su također koristili Lisp i nastavili su ga razvijati prema njihovim potrebama.
Sile koje su gurale razvoj značajki Lispa također su utjecale i na druge pojedinosti – veliki problemi umjetne inteligencije koriste mnogo računalnih resursa, i ako pogledamo Mooreov zakon u zadnjih 30 godina, možete zamisliti koliko su oskudni bili računalni resursi na računalima osamdesetih godina. Tako su korisnici Lispa morali pronalaziti razne načine da iskoriste performanse njihovih implementacija.
Tada je također bila i era Lisp strojeva, s nekoliko tvrtki koje su proizvodile računala koja su nativno pokretala Lisp. Lisp je postao programski jezik za razvoj računalnih sustava, uređivača teksta, jezični prevoditelj, i sve ostalo što je bilo pokretano na Lisp stroju.
Zapravo, ranih osamdesetih godina, s raznim laboratorijima umjetne inteligencije i prodavača Lisp strojeva od kojih su svi prodavali svoje Lisp implementacije, došlo je do naglog širenja Lisp sustava i dijalekta, na što je DARPA reagirala i izrazila zabrinutost o raspadu Lisp zajednice. Kako bi riješili problem, osnovna grupa Lisp hakera se odlučila sastati 1981. godine i započela proces standardizacije jezika koji se zove *Common Lisp* koji je bio kombinacija najboljih značajki od postojećih Lisp dijalekata.

3.2. Matematički temelji Lispa

Lisp je bio prvi funkcionalni jezik prema [26] koji je temeljen na matematičkoj teoriji računanja. Lisp je u osnovi funkcionalni jezik inspiriran lambda računom koji je računalni formalizam temeljen na funkcijama i primjenama funkcija. Ovo je alternativa poznatijem računalnom modelu, Turingovom stroju, koji je mnogo bliže računalima koji se danas koriste.
Ako se zamisli jednostavna funkcija zbrajanja gdje se zbrajaju x i y , koji se često zapisuju poput $x + y$ gdje su x i y varijable koje su zamjenjene vrijednostima koje se žele zbrojiti. Drugi način je gledati na + kao na funkciju dva argumenta + (x, y); kada se želi zbrojiti, npr. 3 i 4 tada se to može zapisati kao + (3, 4) i kada se obradi funkcija kao povratnu vrijednost dobije se broj 7, kao suma x i y. Ovu ideju se može generalizirati i definirati bilo koju funkciju f s poznatim brojem argumenata, $f(x_1,x_2,...,x_n)$. Ovdje, $x_1,...,x_n$ se nazivaju formalni argumenti i kada se zapravo želi obraditi funkcija f tada se pružaju „stvarni argumenti“ (kao npr. 3 i 4 za + u prethodnom primjeru). Ovakve vrste fukcija su dio većine programskih jezika u jednom ili drugom obliku.
Zanimljivo zapažanje je vidjeti funkciju dva argumenta kao jedan argument gdje ako mu se pruži samo jedan stvarni argument onda se kao odgovor dobije funkcija. Ako se same funkcije smatraju vrijednostima koje se mogu dobiti kao odgovor tada se može izgraditi računalni model s funkcijama koje imaju samo jedan argument.
Alonzo Church [27] je formalizirao konstrukciju funkcija i primjene funkcija sustavno kroz λ-račun. Na primjer, ako se pretpostavi da je + primitivna funkcija tada se prethodni primjer može zapisati kao $\lambda x.\lambda y.\ +\ x\ y$ gdje λ označava x i y kao formalne argumente.
Lisp daje jezičnu dimenziju λ-računalnom modelu s nekim promjenama i dodacima tako da je njegova primjena praktičnija.
Lisp zamjenjuje λ-izraze sa S-izrazima (simboličnim izrazima). S-izrazi su definirani jednako kao i λ-izrazi. Također McCarthy definira M-izraze (meta izraze) koji sadrže S-funkcije koje djeluju na S-izraze. Postoji pet primitivnih S-funkcija i dva funkcionalna oblika, kondicional i rekurzija. Kao dodatak, kroz mehanizam „oznaka“, funkcionalni oblik može biti imenovan i pozvan rekurzivno unutar njegovog M-izraza.
S-izrazi su definirani s naredna dva pravila:

- Svaki atomski simbol je S-izraz
- Ako su S_1 i S_2 S-izrazi onda je (S_1.S_2) također S-izraz

S-izraz poput (A.(B.(C.(D.NIL)))) su često napisani u obliku liste (A, B, C, D). NIL je posebni atomski simbol koji završava liste. Najčešće se koriste liste umjesto S-izraza jer su jednostavnije za čitanje i pisanje.

Za početak se trebaju definirati pet primitivnih funkcija S-izraza. To su **car**, **cdr**, **cons**, **atom**, **eq**. Prve dvije se koriste za rastavljanje S-izraza, treća se koristi za konstrukciju ili sastavljanje S-izraza i zadnje dvije su predikati. Treba primjetiti da su zadnje dvije djelomične funkcije i mogu vratiti nedefiniranu vrijednost ukoliko su argumenti krivi.

Primitivne funkcije imaju sljedeća značenja:

- **car** vraća prvi element S-izraza ili liste.
  Primjer: (car (A.B)) je A, (car(A,B,C)) je također A.
- **cdr** daje drugi element S-izraza ili sve elemente liste osim prvog elementa.
  Primjer: (cdr (A.B)) je B, (cdr(A,B,C)) je lista (B,C).
- (**cons** A B) konstruira S-izraz (A.B). Slično, (cons A (B,C)) konstruira listu (A,B,C) i (cons A NIL) vraća listu (A).
- (**atom** S_1) je istinit (T) ukoliko je S_1 atomski S-izraz i laž ukoliko je suprotno (F).
  Primjer: (atom A) je istinit, ali (atom (A.B)) ili (atom (A,B)) su laži.
- (**eq** S_1 S_2) je definiran ako i samo ako su S_1 i S_2 su atomski inače je nedefiniran. Istinito je ako su identični atomi, inače je laž.
  Primjer: (eq A B) je laž, (eq A A) je istina, (eq (A.B) A) ili (eq (A B) B) daju neodređenu vrijednost.

3.3. AutoLISP

AutoLISP je prema [28] programski jezik dizajniran za proširenje i prilagođavanje funkcionalnosti AutoCAD-a. On je temeljen na LISP programskom jeziku, čije podrijetlo datira iz kasnih pedesetih godina.

AutoLISP je predstavljen kao sučelje za programiranje aplikacija (API) u AutoCAD inačici 2.1., sredinom osamdesetih godina. LISP je odabran kao početni AutoCAD API jer je bio pogodan za proces nestrukturiranog dizajna AutoCAD projekata, koji je uključivao stalno ponavljanje različitih rješenja za dizajn problema.

Razvoj AutoLISP programa za AutoCAD vrši se pisanjem koda u uređivač teksta, a zatim učitavanje koda u AutoCAD i njegovo pokretanje. Otkrivanje grešaka programa se obavlja tako da se dodaju izjave za ispis sadržaja varijabli na strateške točke u programu. Korisnik sam mora otkriti gdje postaviti te točke, te na koje varijable se treba pripaziti. Ako korisnik otkrije da nema dovoljno informacija da otkrije grešku, on se mora vratiti i promjeniti kod dodavajući još točaka za otkrivanje grešaka. Na kraju, kada program se ispravno izvrši, kod za otkrivanje grešaka se treba obrisati ili komentirati.

3.3.1. AutoLISP izrazi

AutoLISP program se sastoji od serije izraza. AutoLISP izrazi imaju sljedeći oblik:

(funkcija argumenti)

Svaki izraz započinje otvorenom (lijevom) zagradom i sastoji se od imena funkcije i neobaveznim argumentom za tu funkciju. Svaki argument može također biti izraz. Izrazi završavaju desnom zagradom. Svaki izraz vraća vrijednost koju mogu koristiti okolni izrazi. Vrijednost zadnjeg prevedenog izraza se vraća u izraz koji je pozvao.

Na primjer ovaj kod ima 3 funkcije:

(fun1 (fun2 argumenti) (fun3 argumenti))

Funkcija fun1 ima dva argumenta, a funkcije fun2 i fun3 imaju po jedan argument. Rezultati funkcija fun2 i fun3 se vraćaju nadređenoj funkciji fun1 koja ovisno o funkciji vraća odgovarajući rezultat.

3.3.2. AutoLISP tipovi podataka

Cjelobrojni tip podataka

Cjelobrojni tip podataka ne sadrži decimalnu točku. Cijeli brojevi u AutoLISP-u su 32 bitni brojevi s predznakom. Kada se eksplicitno koristi cijeli broj u AutoLISP izrazu, tada se ta vrijednost naziva konstantom. Ako se upiše broj veći od maksimalnog cijelog broja onda se vrijednost pretvori u realni broj. Ako je rezultat aritmetičke operacije dvaju cijelih brojeva veći od najvećeg cijelog broja onda će vrijednost biti neispravna.

Realni brojevi

Realni broj je broj koji sadrži decimalnu točku. Brojevi između -1 i 1 moraju sadržavati vodeću nulu. Realni brojevi se spremaju kao brojevi dvostruke preciznosti s pomičnim zarezom, pružajući preciznost od barem 14 značajnih znamenki. Realni brojevi se mogu prikazati u znanstvenom zapisu.

Stringovi

String je skup znakova unutar navodnika. Unutar navodnika znak obrnute kose crte (\) omogućuje kontrolnim znakovima da budu uključeni. Kada se eksplicitno koristi string pod navodnicima u AutoLISP izrazu, ta vrijednost je poznata kao string konstanta.

Primjer: "Rijec 1" ili "\nRijec 1"

Liste

Liste u AutoLISPu su grupe povezanih vrijednosti odvojenih razmakom i okružene zagradama. Liste omogućuju učinkovitu metodu spremanja brojnih povezanih vrijednosti. AutoCAD izražava 3D točke pomoću liste od tri realna broja.

Primjer: (1.0 2.0 3.0), (1 "Jedan")

Selection Sets

Selection Sets su grupe od jednog ili više objekta (entiteta). Korisnik može interaktivno dodati objekte u, ili obrisati objekte iz, Selection Set s AutoLISP rutinama.

Sljedeći primjer koristi ssget funkciju da vrati selection set koji sadrži sve objekte na crtežu.

Primjer: (ssget "X")

Imena entiteta

Ime entiteta je numerička oznaka dodijeljena objektu na crtežu. To je zapravo pokazivač na datoteku koju održava AutoCAD, i može se koristiti da se pronađe zapis o objektu u bazi podataka i njegovi vektori. Ova oznaka može biti referencirana od strane AutoLISP funkcija da omogući označavanje objekata za obradu na razne načine. U AutoCAD-u se entiteti odnose na objekte. Funkcija entlast vraća ime posljednjeg objekta napravljenog na crtežu.

3.3.3. AutoLISP varijable

AutoLISP varijabla pretpostavlja tip podatka koji joj je dodijeljen. Dok im se ne dodijeli nova vrijednost, varijable zadržavaju svoju originalnu vrijednost. Za dodijeljivanje vrijednosti varijablama koristi se setq funkcija.

Primjer: (setq varijabla1 123)

Za prikaz vrijednosti varijable u AutoCAD komandnom sučelju, potrebno je staviti uskličnik ispred imena varijable.

Primjer: !varijabla1

Nil varijable

AutoLISP varijable koje nemaju dodijeljenu vrijednost imaju vrijednost nil. Ovo je drugačije od blank, koji se smatra string znakom, i drugačiji je od nule, koja je broj. Kod provjeravanja trenutne vrijednosti varijable, može se provjeriti i je li varijabli dodijeljena vrijednost.

Svaka varijabla koristi malu količinu memorije, tako da je dobro prilikom programiranja ponovno koristiti imena varijabli ili ih postaviti na nil kada više nisu potrebne. Postavljanjem varijable na nil oslobađa se memorija.

Primjer: (setq varijabla1 nil)

Predefinirane varijable

Sljedeće predefinirane varijable se često koriste u AutoLISP aplikacijama:

PAUSE – definira se kao string koji sadrži \\ znak. Ova varijabla se koristi s *command* funkcijom za pauziranje za korisnikov unos.

PI – definira se kao konstanta p (pi). Vrijednost je 3.14159.

T – definira se kao konstanta T. Koristi se kao *non-nil* vrijednost.

3.3.4. Rukovanje brojevima i stringovima

AutoLISP pruža funkcije za rad s cijelim i realnim brojevima. Kao dodatak kod izvođenja složenih matematičkih izračuna u aplikaciji, korisnik može koristiti funkcije za rukovanje brojevima kao pomoć za svakodnevno korištenje AutoCAD-a. Ako su sve vrijednosti cijeli brojevi rješenje će biti cijeli broj, ali ako je jedna od vrijednosti realan broj rezultat će biti realan. Neke od aritmetičkih funkcija su: +, -, *, /, 1+, 1-, abs, exp, float, sin, cos, log, sqrt.
Primjer: (* 2 3)

AutoLISP pruža funkcije za rad sa string vrijednostima. Na primjer, **strcase** pretvara sve znakove abecede u velika ili mala slova. On prima dva argumenta, string i neobavezan argument koji određuje hoće li slova biti velika ili mala. Ukoliko nije postavljen drugi argument on se smatra nil i time funkcija strcase vraća string s velikim slovima.
Primjer: (strcase "Ovo je TEST." T) Rezultat: "ovo je test."

Funkcija **strcat** kombinira više stringova u jednu string vrijednost. Ovo je korisno kod postavljanja varijabilnog stringa unutar konstantnog string.
Primjer: (setq string "crna") (setq string2 (strcat "Ovo je" string "macka."))
Rezultat: "Ovo je crna macka."

Funkcija **strlen** vraća broj znakova koji se nalaze u stringu.
Primjer: (strlen string) Rezultat: 4

3.3.5. Rukovanje listama

AutoLISP pruža funkcije za rad s listama. Liste pružaju efikasnu i snažnu metodu spremanja brojnih povezanih vrijednosti. Ipak, LISP je nazvan jezikom za rad s listama. Kada korisnik razumije snagu lista, on može kreirati snažnije i fleksibilnije aplikacije.
Nekoliko AutoLISP funkcija pruža temelje za programiranje dvodimenzionalnih i trodimenzionalnih grafičkih aplikacija. Ove funkcije vraćaju vrijednosti točaka u obliku liste.
Funkcija **list** pruža jednostavnu metodu grupiranja povezanih stavki.
Primjer: (setq nekalista (list 1 "broj" 2.4)

Ukoliko se želi dobiti vrijednost određene stavke iz liste nekalista onda možemo koristiti **nth** funkciju. Ova funkcija prima dva argumenta. Prvi argument je cijeli broj koji označava koju stavku dohvatiti. 0 označava prvu stavku u listi, 1 označava drugu stavku u listi, itd. Drugi argument je sama lista.

Primjer: (nth 2 nekalista) – ovaj primjer vraća vrijednost 2.4

Funkcija **cdr** vraća sve elemente liste osim prvog elementa, dok **car** funkcija vraća samo prvi element liste.

Primjeri: (setq lista2 (1 2 3 4 5))

(car lista2) – vraća vrijednost 1

(cdr lista2) – vraća vrijednosti 2 3 4 5

(cadr lista2) – vraća vrijednost 2

Funkcija **cons** vraća listu s novom stavkom dodanom na početku liste, dok **append** funkcija vraća listu s novim stavkama dodanim na kraj liste. Važno je napomenuti da **append** radi isključivo s listama tako da svi argumenti ove funkcije moraju biti liste. Dodavanjem jednostrukih navodnika (') na jednostavan način se pretvori string u listu.

Primjeri: (setq lista3 (append lista2 '("Jedan"))) – Rezultat je (1 2 3 4 5 "Jedan")

(setq lista4 (cons "Dva" lista3)) – Rezultat je ("Dva" 1 2 3 4 5 "Jedan")

Liste točaka

AutoLISP koristi sljedeće konvencije za rukovanje grafičkim koordinatama. Točke su izražene kao liste od dva ili tri broja okružena zagradama.

2D točke su izražene kao liste koje sadržavaju dva realna broja (X i Y), kao npr. (2.3 8.42)

3D točke su izražene kao liste koje sadržavaju tri realna broja (X, Y i Z), kao npr. (4.3 2.35 8.2)

Za dodjeljivanje koordinata točkama može se koristiti izraz: (setq tocka1 (list 2.34 5.42)), te ukoliko su vrijednosti konstante na jednostavan način način se mogu pretvoriti u listu npr. (setq tocka1 '(2.34 5.42)).

3.3.6. Definiranje funkcije

S AutoLISP-om se mogu definirati vlastite funkcije. Jednom definirane, ove funkcije se mogu koristiti u AutoCAD-ovom komandnom sučelju., Visual LISP konzoli, ili unutar AutoLISP izraza, baš kao što se koriste i standardne fukcije. Također se mogu kreirati i vlastite AutoCAD naredbe, zato jer su naredbe samo posebna vrsta funkcija.

Funkcija **defun** kombinira grupu izraza u funkciju ili naredbu. Ova funkcija zahtjeva barem tri argumenta, gdje je prvi argument ime definirane funkcije. Drugi argument je lista argumenata (lista argumenata i lokalnih varijabli koje funkcija koristi). Lista argumenata može biti nil ili prazna lista ( ) . Iza ovih argumenata su izrazi koji prave funkciju; mora biti barem jedan izraz u definiciji funkcije.

Primjer: (defun pozdrav ( ) (princ "Dobar dan")(princ))

C:XXX funkcije

C:XXX značajka se može koristiti za dodavanje novih naredbi u AutoCAD ili ponovno definiranje postojećih naredbi.

Za korištenje funkcija kao AutoCAD naredbe, moraju se poštivati ova pravila:

- Ime funkcije mora imati oblik C:XXX (velika ili mala slova). C: dio uvijek mora biti prisutan; XXX dio je ime naredbe po izboru..
- Funkcija mora biti definirana bez argumenata. Ipak, lokalne varijable su dozvoljene i dobra ih je praksa koristiti.

Primjer: (defun C:HELLO () (princ "Hello world.")(princ))

Naredba: hello

Rezultat: Hello world.

Korištenje lokalnih varijabli

U sljedećem primjeru će biti prikazano korištenje lokalnih varijabli u korisnički definiranoj funkciji (treba biti razmak barem jedno mjesto između kose crte i lokalne varijable).

Primjer:

```
(defun LOKALNA (/ aa bb)
(setq aa "A" bb "B")
(princ (strcat "aa ima vrijednost" aa))
(princ (strcat "bb ima vrijednost" bb))
(princ))
```

Funkcije s argumentima

U AutoLISP-u, korisnik može definirati funkcije koje primaju argumente. Za razliku od mnogih standardnih AutoLISP funkcija, korisnički definirane funkcije ne mogu primiti neobavezne argumente. Kada se pozove korisnički definirana fukcija koja prima argumente, korisnik mora pružiti vrijednost za sve argumente.

Argumenti se tretiraju kao posebna vrsta lokalne varijable; argumenti nisu dostupni izvan funkcije. Funkcija ne može sadržavati više argumenata istog imena.

Sljedeći primjer će prikazati funkciju koja prima dva string argumenta i kombinira ih s još jednim stringom i kao rezultat vraća novi string.

Primjer:

```
(defun STR ( arg1 arg2 / aaa)
(setq aaa "Neki string")
(strcat aaa "," arg1 "," arg2))
(setq novistring (STR "string 1" "string 2"))
```

Rezultat: "Neki string, string 1, string 2"

3.3.7. Petlje

While

Sintaksa: (while izraz radiovo)

Izraz – bilo koji AutoLISP izraz koji ima vrijednost true ili ne nil

Radiovo – bilo koji valjan AutoLISP izraz ili izrazi

Primjeri: (while T (princ 1)) – beskonačna petlja koja ispisuje vrijednost 1

(while nil (princ 1)) – izlazak iz petlje

Primjer brojača:

```
(setq brojac 1)
(while (<  brojac 4)
      (princ brojac)
      (setq brojac (+ brojac 1))
 )
```

Repeat

Sintaksa: (repeat integer)

Integer – bilo koji cijeli broj.

Primjer: (repeat 20 (princ "A"))

Rezultat: Ispis slova A 20 puta

If

Sintaksa: (if OvoJeIstinito NapraviOvo)

(if OvoJeIstinito NapraviOvo InačeNapraviOvo)

(if OvoJeIstinito (progn NapraviSveOvo) (progn InačeNapraviSveOvo)

(if OvoJeIstinito (progn NapraviSveOvo) InačeNapraviOvo)

(if OvoJeIstinito NapraviOvo (progn InačeNapraviSveOvo)

progn – znači da će biti više od jedne izjave

Primjeri: (if T (princ 1)) – ispisuje broj 1

(if T (princ 1) (princ 2)) – ispisuje broj 1 jer T predstavlja istinu

(if nil (princ 1) (princ 2)) – ispisuje broj 2 jer nil predstavlja neistinu

---

```
(if (= 3 3)
    (progn (princ 1) (princ 2))
    (progn (princ 3) (princ 4))
)
```

Rezultat: Ispisuje se vrijednost 1 i 2 .

3.3.8. Korištenje AutoLISP-a za komunikaciju s AutoCAD-om

AutoLISP sadrži razne funkcije za ispitivanje sadržaja trenutno učitanog crteža. Ovo poglavlje upoznaje korisnika s tim funkcijama i opisuje kako ih koristiti zajedno s drugim funkcijama.

Funkcije **query** i **command** opisane u ovom poglavlju pružaju direktni pristup AutoCAD naredbama i servisima crteža. Njihovo ponašanje ovisi o trenutnom stanju AutoCAD sustava i varijablama okoline, te o trenutno učitanom crtežu.

Funkcija **command** šalje AutoCAD naredbu direktno AutoCAD komandnom sučelju. Funkcija **command** ima promjenjivu duljinu liste argumenata. Ovi argumenti moraju odgovarati tipu i vrijednostima očekivanih od komandnog sučelja; oni mogu biti *string*, realne vrijednosti, cjelobrojne vrijednosti, točke, imena entiteta, ili imena skupa odabira. Podaci poput kuteva, udaljenosti, i točaka mogu biti prosljeđeni u *string* ili kao vlastite vrijednosti (cjelobrojne vrijednosti, realne vrijednosti ili kao liste točaka). Prazan *string* (" ") je ekvivalentan pritisku na razmaknicu ili *Enter* na tipkovnici.

Postoje neka ograničenja na naredbe koje se mogu koristiti s **command** funkcijom.

Naredni dio koda predstavlja reprezentativne pozive **command** funkcije.

Primjer:

```
(command "circle" "0,0" "3,3")
(command "thickness" 1)
(setq p1 '(1.0 1.0 3.0))
(setq rad 4.5)
(command "circle" p1 rad)
```

Ukoliko je AutoCAD u komandnom sučelju kada se pozovu ove funckuje AutoCAD će izvršiti sljedeće akcije:

1. Prvi poziv funkcije **command** prosljeđuje točke naredbi **CIRCLE** kao *string* (crta kružnicu sa središtem u 0.0, 0,0 i prolazi kroz 3.0, 3.0)
2. Drugi poziv prosljeđuje cjelobrojnu vrijednost u **THICKNESS** sistemsku varijablu (mijenja trenutnu debljinu u 1.0)
3. Zadnji poziv koristi 3D točku i realnu vrijednost, od kojih su obje spremljene kao varijable i prosljeđene od reference naredbi **CIRCLE**.

3.2.9. Sistemske varijable i varijable okoline

Pomoću **getvar** i **setvar** funkcija, AutoLISP aplikacije mogu istražiti i promjeniti vrijednost AutoCAD sistemskih varijabli. Ove funkcije koriste *string* kako bi odredile ime varijable. Funckija **setvar** određuje vrijednost tipa kojeg sistemska varijabla očekuje. AutoCAD sistemska varijabla dolazi u raznim tipovima: cjelobrojni tip, realne vrijednosti, *string*, 2D točke, i 3D točke. Vrijednosti isporučene kao argumenti funkciji **setvar** moraju biti očekivanog tipa. Ako je isporučen krivi tip podatka, AutoLISP ispije da je generirana pogreška.

Primjer:

Sljedeći dio koda osigurava da je sljedeća **FILLET** naredba najmanjeg radijusa 1.

```
( if (< (getvar "filletrad") 1)
        (setvar "filletrad" 1)
)
```

3.2.10. Dobivanje korisničkog unosa

Svaki korisnički unos **getxxx** funkcije pauzira se za unos podataka određenog tipa i vraća unešenu vrijednost. Aplikacija određuje izborni upit za prikaz prije nego se funkcija pauzira. Sljedeća lista prikazuje **getxxx** funkcije i koji tip korisničkog unosa je potreban.

Tablica 3.1.: Tablica getxxx funkcija i pripadajućih tipova za unos podataka

Ime funkcije	Tip koriničkog unosa
getint	Cjelobrojna vrijednost na komandnoj liniji
getreal	Realna ili cjelobrojna vrijednost na komandnoj liniji
getstring	*String* na komandnoj liniji
getpoint	Vrijednost točke na komandnoj liniji ili označena na zaslonu
getcorner	Vrijednost točke na komandnoj liniji ili označena na zaslonu
getdist	Realna ili cjelobrojna vrijednost (ili udaljenost) na komandnoj liniji ili označena na zaslonu
getangle	Vrijednost kuta (u trenutnom formatu kuta) na komandnoj liniji ili s obzirom na označene točke na zaslonu
getorient	Vrijednost kuta (u trenutnom formatu kuta) na komandnoj liniji ili s obzirom na označene točke na zaslonu
getkword	Predefinirana ključna riječ ili skraćenica na komandnoj liniji

3.2.11. Tekstualne mjere

Funkcija **textbox** vraća dijagonalne koordinate kutije koja ograđuje tekstualni objekt. Potrebna je lista definicija entiteta povratnog tipa funkcije **entget** (lista asocijacija grupnih kodova i vrijednosti) kao jedini argument. Ova lista može sadržavati cijeli opis liste asocijacija tekstualnog objekta ili samo listu koja opisuje tekstualni *string*.

Točke vraćene od **textbox** opisuju kutiju povezivanja (imaginarnu kutiju koja zatvara tekstualni objekt) tekstualnog objekta, kao da je točka umetanja pozicionirana na (0,0,0) i njezin kut rotacije je 0. Prva vraćena lista je točka (0.0 0.0 0.0), ukoliko tekstualni objekt je nakrivljen, vertikalan ili sadrži slova s padobranima (poput g i p). Vrijednost prve liste točaka označava udaljenost odmaka od točke umetanja teksta do donjeg lijevog ugla najmanjeg pravokutnika koji obavija tekst. Druga lista točaka određuje gornji desni ugao pravokutnika te

kutije. Povratna lista točaka uvijek opisuje donji lijevi i gornji desni ugao kutije povezivanja, neovisno o orijentaciji teksta.

Sljedeći primjer prikazuje minimalnu dozvoljenu listu definicije entiteta koju **textbox** prihvaća. Jer nisu pružene dodatne informacije, textbox koristi trenutne početne vrijednosti za izgled i veličinu teksta.

Primjer:

Naredba: (textbox '((1 . "*Hello world*")))

((0.0 0.0 0.0) (2.80952 1.0 0.0))

Fotografija 3.1.: Prikaz riječi *Sample* u textboxu

3.2.12. Rukovanje skupom odabira

AutoLISP pruža mnogo funkcija za rukovanje skupom odabira. Funkcija **ssget** pruža najopćenitiji način za kreiranje skupa odabira. Ona može kreirati skup odabira na sljedeće načine:

- Eksplicitno određivanje koje objekte označiti koriseći (*Last*, *Previous*, *Window*, itd.)
- Određivanjem jedne točke
- Označavanjem cijele baze podataka
- Tražeći od korisnika da označi objekte

Sa svakom opcijom, može se koristiti filtriranje za određivanje liste atributa i uvjeta koje označeni objekti moraju zadovoljiti.

Prvi argument naredbe **ssget** je *string* koji opisuje koju opciju odabira koristiti. Sljedeća dva argumenta, pt1 i pt2, određuju vrijednost točaka za relevantne opcije. Lista točaka, *pt-list,* mora biti pružena kao argument metodama odabira koje omogućuju odabir mnogokuta. Zadnji argument, lista filtera, je neobavezna.

Tablica 3.2.: Primjeri korištenja naredbe ssget

Funkcijski poziv	Rezultat
(setq pt1 '(0.0 0.0 0.0) pt2 '(5.0 5.0 0.0) pt3 '(4.0 1.0 0.0) pt4 '(2.0 6.0 0.0))	Dodjeljuje vrijednosti točkama pt1, pt2,pt3 i pt4
(setq ss1 (ssget))	Korisnik se pita za opći odabir objekata i postavlja te vrijednosti u skup odabira
(setq ss1 (ssget "P"))	Kreira se skup odabira iz najnovijeg skupa odabira
(setq ss1 (ssget "L"))	Kreira se skupa odabira zadnjeg objekta dodanog u bazu podataka koji je vidljiv na zaslonu
(setq ss1 (ssget pt2))	Kreira se skup odabira koji prolazi kroz točki (5,5)
(setq ss1 (ssget "W" pt1 pt2))	Kreira se skup odabira objekata koji su unutar prozora od (0,0) do (5,5)
(setq ss1 (ssget "X"))	Kreira se skup odabira objekata za sve objekte unutar baze podataka

3.2.13. Testiranje odnosa

Ukoliko nije drugačije označeno , jednakost je obuhvaćena za svaki predmet u listi filtera. Za numeričke grupe (cjelobrojni, realni, točke, vektori), mogu se odrediti drugi odnosi uključujući specijalni -4 grupni kod koji određuje relacijski operator. Vrijednost -4 grupe je *string* koji ukazuje na test operator koji će se primjenjivati na sljedeću grupu u filter listi.
Sljedeći primjer označava sve kružnice u promjeru (grupni kod 40) koji je veći ili jednak 2.0.
Primjer: (ssget "X" '((0 . "CIRCLE") (-4 . ">=") (40 . 2.0)))

Tablica 3.3.: Opis relacijskih operatora

Operator	Opis
"!="	Nije jednako
"/="	Nije jednako
"<>"	Nije jednako
"<"	Manje od

"<="	Manje ili jednako od
">"	Veće od
">="	Veće ili jednako od
"="	Jednako
"*"	Bilo koja vrijednost (uvijek točno)

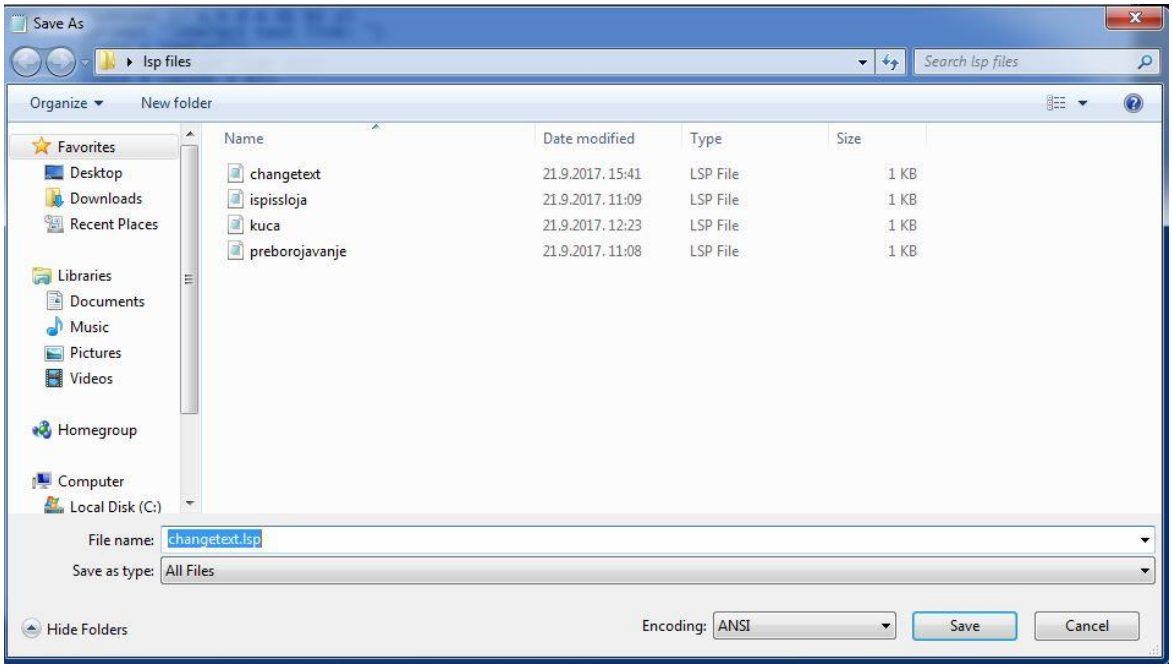
Korištenje relacijskih operatora ovisi o vrsti grupa koje se testiraju:

- Svi relacijski operatori osim operatora bitova ("&" i "&=") se mogu koristiti i za cjelobrojne i realne grupe
- Za grupe točaka, X, Y, i Z testovi mogu biti kombinirani u jedan string, sa svakim operatorom odvojenim zarezom (npr. ">, >, * "). Ukoliko je operator izostavljen iz *stringa* (npr. "=,<>" onda ostavlja Z bez testa), onda je "bilo koja je vrijednost" operator, "*", pretpostavljen
- Vektori smjera (grupa tipa 210) može biti uspoređena jedino s operatorima " * ", "=", i "!=" (ili jednim koji odgovara "nije jednako" *stringu*)

3.2.14. Otvaranje LISP dokumenta u AutoCAD-u

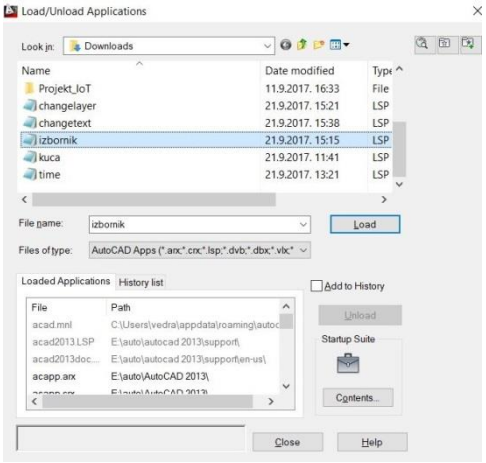
1. Prvi korak je napisati LISP kod u programskom alatu *Notepad*.
2. Zatim je potrebno spremiti taj dokument s nastavkom .lsp . Tip podataka treba biti *All Files*, a enkodiranje ANSI.

Fotografija 3.2.: Spremanje LISP datoteke



3. Zatim je potrebno u AutoCAD-u upisati naredbu APPLOAD u komandno sučelje.
4. Nakon što se naredba izvrši treba se otvoriti prozor kao na fotografiji 3.3. gdje se odabere tražena datoteka i učita u AutoCAD.

Fotografija 3.3.: Učitavanje LISP datoteke u AutoCAD



5. Pokrenite svoju datoteku pokretanjem funkcije iz LISPa. (defun c:**funkcija** ())

4. Primjeri AutoLISP modula za AutoCAD

U ovom poglavlju će biti prikazani primjeri korištenja AutoLISP modula u AutoCAD-u. Svaki primjer je ukratko objašnjen.

4.1. Lisp modul koji prebrojava više po volji odabranih korisničkih objekata i eksportira ih u .txt datoteku

Na početku se definira funkcija **objekti** koja je zapisana u obliku naredbe. Zatim korisnik pomoću **ssget** funkcije označi objekte na zaslonu. Funkcija **sslength** uzima broj označenih objekata iz *Selection Set* i vraća cijeli broj.
Drugi dio koda pravi novu tekstualnu datoteku pomoću **getfiled** funkcije. Prvi argument **getfiled** prima naslov pri unosu imena datoteke. Drugi argument prima putanju koja je u ovom slučaju *default*, treći argument prim tip datoteke i četvrti argument označava opcije. **Open** funkcija otvara tu tekstualnu datoteku i označava pomoću "w" opcije da će se u tu datoteku pisati. **Princ** funkcija piše u datoteku i na kraju se pomoću **close** funkcije datoteka zatvara.

Lisp kod:

```
(defun c:objekti ()
        (setq eset (ssget))
        (setq broj (sslength eset))
        (setq imed (getfiled "Unesite ime tekstualne datoteke" "" "txt" 1))
        (setq fil (open imed "w"))
        (princ "Broj oznacenih objekata je: " fil)
        (princ broj fil)
        (close fil)
)
```

4.2. Lisp modul za ispis imena sloja svakog entiteta unutar *Selection Set*

U ovom primjeru se prvo definira funkcija **sloj**. Zatim se u **if** petlji provjerava jesu li označeni entiteti. Ukoliko nisu ispisuje se greška. Potrebno je koristiti **progn** jer postoji više izjava. Brojač se postavlja na početnu vrijednost 0 koja je ujedno i prvi podatak *Selection Seta*. U while petlji se prolazi kroz sve entitete do **sslength** tj. maksimalne vrijednost označenih objekata. Funkcija **ssname** zatim dohvaća imena svakog entiteta, a funkcija **entget** dohvaća DXF kodove za entitet.
U imesloja se sprema ime sloja. Ispisuje se ime sloja i na kraju se povećava brojač i prelazi se na sljedeći podatak.

Lisp kod:

```
(defun c:sloj()
        (if (setq eset(ssget)
            (progn
                (setq brojac 0)
                (while (< brojac (sslength eset))
                        (setq ent (ssname eset brojac))
                        (setq enlist (entget ent))
                        (setq imesloja (cdr (assoc 8 enlist)))
                        (princ "\n")
                        (princ imesloja)
                        (setq brojac (+ brojac 1))
                )
            )
            (princ "\n Greska – Nema oznacenih entiteta.")
        )
)
```

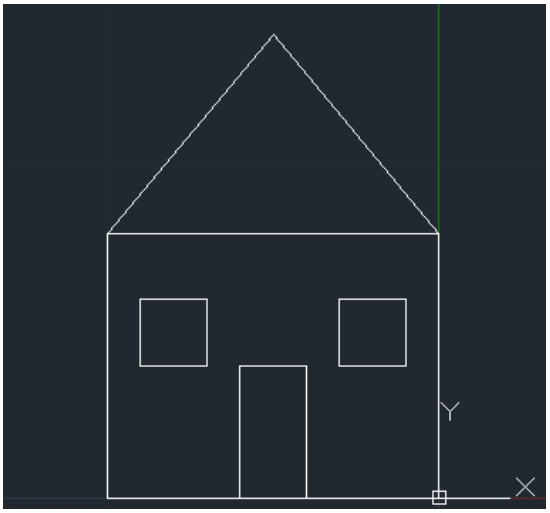
4.3. Lisp modul koji crta kuću

Ovaj primjer prikazuje izradu jednostavne kuće pomoću linija. Na početku se definira funkcija **kuca**. Zatim se točkama p1 – p17 dodjeljuju vrijednosti njihovih koordinata. Naredbom **command "line"** iscrtavaju se dužine koje primaju točke kao parametre.

Lisp kod:

```
(defun c:kuca ()
        (setq pt1 '(0 0) pt2 '(-10 0) pt3 '(-10 8) pt4 '(0 8) pt5 '(-4 0) pt6 '(-6 0)
        pt7 '(-4 4) pt8 '(-6 4) pt9 '(-5 14) pt10 '(-7 4) pt11 '(-9 4) pt12 '(-7 6)
        pt13 '(-9 6) pt14 '(-3 4) pt15 '(-1 4)  pt17 '(-3 6) pt16 '(-1 6))
        (command "line" pt1 pt2 "") (command "line" pt2 pt3 "")
        (command "line" pt3 pt4 "") (command "line" pt1 pt4 "")
        (command "line" pt3 pt9 "") (command "line" pt4 pt9 "")
        (command "line" pt6 pt8 "") (command "line" pt7 pt8 "")
        (command "line" pt5 pt7 "") (command "line" pt10 pt11 "")
        (command "line" pt11 pt13 "") (command "line" pt12 pt13 "")
        (command "line" pt10 pt12 "") (command "line" pt15 pt14 "")
        (command "line" pt14 pt17 "") (command "line" pt16 pt17 "")
        (command "line" pt16 pt15 "")
)
```

Fotografija 4.1.: Prikaz rezultata funkcije kuca



4.4. Lisp modul za iščitavanje datuma i vremena

Na početku se definira funkcija **datum** i deklariraju se lokalne varijable. Zatim se dobije datum pomoću naredbe **getvar "CDATE"** . Nareba **rtos** pretvara datum u *string*, kao drugi argument koji prima koristi se mod 2 koji je decimalan, a treći argument označava preciznost. Naredba **substr** prima 3 argumenta koji su *string*, početno mjesto i duljina *stringa*. U ovom primjeru se iz *stringa* "d" izvlači traženi dan, mjesec i godinu. Na kraju se *stringovi* povežu naredbom **strcat** i ispisuju na ekran. Na slični se način ispisuje i **vrijeme**.

Lisp kod za ispis datuma:

```
(defun c:datum ( / d god mj dan)
   (setq d (rtos (getvar "CDATE") 2 6)
      god (substr d 3 2)
      mj (substr d 5 2)
     dan (substr d 7 2)
    )
   (strcat dan "/" mj "/" god)
)
```

Lisp kod za ispis vremena:

```
(defun c:vrijeme ( / d hr m s)
   (setq d (rtos (getvar "CDATE") 2 6)
      hr (substr d 10 2)
      m (substr d 12 2)
      s (substr d 14 2)
   )
   (strcat hr ":" m ":" s)
)
```

4.5. Lisp modul za promjenu teksta

Ovaj Lisp modul prema [29] zamjenjuje označeni tekst novim tekstom. Prvo se definira funkcija **subtext** u kojoj se deklariraju lokalne varijable. Zatim se ispisuje tekst naredbom **prompt**. U varijablu "a" se dodjeljuje **entsel** koji traži od korisnika da odabere objekt sa zaslona. Funkcija **entget** dohvaća DXF kodove za entitet i sprema prvi član u varijablu "b" . Korisnik upisuje novi tekst pomoću naredbe **getstring** koji će zamjeniti prethodni. Naredba **cons** vraća listu s novim članom dodanim na početak. Dok naredba **subst** zamjenjuje staru vrijednost u listi novom vrijednošću koja je tekst koji smo upisali. Naredba **entmod** modificira listu dodjeljujući novu vrijednost entitetu.

Lisp kod:

```
(defun c:subtext (/ a b d e d1 b1 y)
        (prompt "\nOdaberi tekst: ")
        (setq a (entsel))
        (setq b (entget (car a)))
        (setq d (assoc 1 b))
        (prompt (cdr d))(terpri)
        (setq e (getstring 1))
        (setq d1 (cons (car d) e))
        (setq b1 (subst d1 d b))
        (entmod b1)
        (princ)
)
```

4.6. Lisp modul za izradu pravilnog mnogokuta

Ovaj modul omogućuje crtanje bilo kojeg pravilnog mnogokuta određivanjem broja stranica i površine. Na početku se definira funkcija **pm** u kojoj se deklariraju lokalne varijable. Zatim korisnik upisuje broj stranica mnogokuta i polumjer unutarnje kružnice mnogokuta. Modul nakon unosa vrijdenosto izračunava kut i apotemu prema formuli 4.1. U AutoLISP-u se ne može koristiti funkcija tan tako da se funkcija tan zapisuje pomoću funkcija **sin** i **cos**. Na kraju korisnik označava središte mnogokuta i pomoću naredbe **POLYGON** se iscrtava mnogokut.

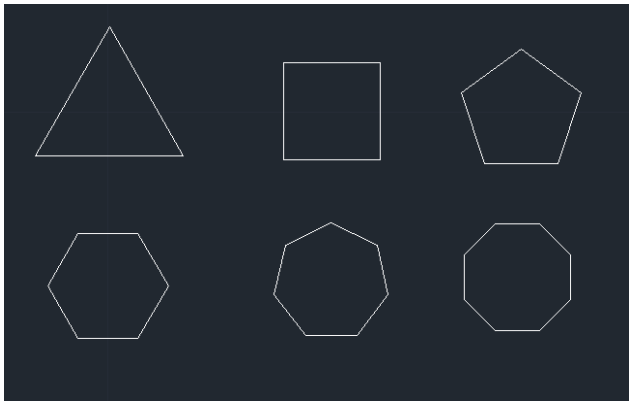
Formula 4.1.: Formula za izračun apoteme

$$A = \sqrt{\frac{Povrsina}{N\ tan\frac{\pi}{N}}}$$

Lisp kod:

```
(defun c:pm (/ a n apt ptloc)
        (setq n (getint "Broj stranica mnogokuta: "))
        (setq a (getreal "Ocekivana povrsina mnogokuta: "))
        (setq ang (/ pi n))
        (setq apt (sqrt (/ (/ a (/ (sin ang) (cos ang))) n))) ;
        (setq ptloc (getpoint "Odaberite poziciju: "))
        (command "_POLYGON" n ptloc "C" apt)
)
```

Fotografija 4.2.: Prikaz rezultata funkcije pm



5. Zaključak

CAD sustavi su razvojem tehnologije jako napredovali. Njihova primjena je u gotovo svim industrijskim granama. Uz njihovu pomoć danas je moguće bilo koji proizvod brzo napraviti i prikazati. Također izrada tehničkih crteža nikad bila lakša jer se može lako manipulirati objektima.

LISP s druge strane dobro upotpunjava nedostatke samog *CAD*-a. Ukoliko su potrebni složeni izračuni površina objekata, kutevi, udaljenost među objetima i slično, LISP to može riješiti jednostavno i brzo. Poznavanje LISP-a također pomaže i kod pisanja u drugim programskim jezicima zbog njegove strukture.

U prikazanim primjerima može se uočit kako se mogu izvući podaci o objektima. Svaki objekt u AutoCAD-u ima svoje parametre koji opisuju njegov položaj, boju, vrstu, sloj u kojem se nalaze i slično. LISP kao jezik za rad s listama baš tu briljira jer se ti podaci mogu grupirati u liste. Pomoću par funkcija mogu se izvući željeni podaci te se može manipulirati s njima. Zbog proširenosti *CAD*-a smatram da je znanje LISP-a potrebno za efikasno korištenje *CAD* aplikacije. Razvojem LISP-a posao dizajnera će se znatno olakšati zbog velikih mogućnosti samoga LISP-a.

LITERATURA

[1] Computer Aided Design and Manufacturing, M.M.M. Sarcar, K. Mallikarjuna Rao, K. Lalit Nryan, 2008.

[2] Computer Aided Design, Dr. Nicos Bilalis, Technical University of Crete, 2000., (http://dl.icdst.org/pdfs/files/ed22bd8e464c6ad75ff352f0d14f83c6.pdf), pristup ostvaren 20.6.2017.

[3] CAD/CAM Theory & Practice, Ibrahim Zeid,1991., (https://books.google.hr/books?id=DJeRtGVAr6MC&pg=PA20), pristup ostvaren 21.1.2018.

[4] W. L. Hosch , Ivan Edward Sutherland, (https://www.britannica.com/biography/Ivan-Edward-Sutherland), pristup ostvaren 13.2.2018.

[5] I. E. Sutherland, Sketchpad: A man-machine graphical communication system, (http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-574.pdf), pristup ostvaren 13.2.2018.

[6] P. J. Schneider, NURB Curves: A Guide for the Uninitiated, (http://www.mactech.com/articles/develop/issue_25/schneider.html), pristup ostvaren 20.1.2018.

[7] Wikipedia, Class A surface, (https://en.wikipedia.org/wiki/Class_A_surface), pristup ostvaren 20.1.2018.

[8] CATIA, Sheet Metal Design, (https://www.cadcam-group.eu/hr/proizvodi/catia-sheet-metal-dizajn), pristup ostvaren 20.1.2018.

[9] Tehničko crtanje i CAD, Nikola Klem, Željko Koški, Irena Ištoka Otković, Osijek, 2010.

[10] progeCAD, (http://www.progecad.com.hr/cd/46/prirucnik-za-progecad-na-hrvatskom-i-engleskom), pristup ostvaren 20.3.2018.

[11] Common Lisp, Kent M. Pitman, 1993. , (https://www.csee.umbc.edu/courses/331/resources/papers/Brief-History-of-Lisp.pdf), pristup ostvaren 10.6.2017

[12] Stanford, Professor John McCarthy,(http://jmc.stanford.edu/), pristup ostvaren 20.1.2018.

[13] Wikipedia, L. Peter Deutsch, (https://en.wikipedia.org/wiki/L._Peter_Deutsch), pristup ostvaren 20.1.2018.

[14] Wikipedia, Daniel G. Bobrow, (https://en.wikipedia.org/wiki/Daniel_G._Bobrow), pristup ostvaren 20.1.2018.

[15] Wikipedia, Richard Greenblatt, (https://en.wikipedia.org/wiki/Richard_Greenblatt_(programmer)), pristup ostvaren 20.1.2018.

[16] Xerox, (https://www.xerox.com/), pristup ostvaren 20.1.2018.

[17] S. E. Fahlman, Autobiography, (https://www.cs.cmu.edu/~sef/), pristup ostvaren 20.1.2018.
[18] W. Bader, VAX, (https://williambader.com/museum/vax/vaxhistory.html), pristup ostvaren 20.1.2018.
[19] Wikipedia, Richard P. Gabriel, (https://en.wikipedia.org/wiki/Richard_P._Gabriel), pristup ostvaren 20.1.2018.
[20] G. Bell, LLNL S1 Mark IIA supercomputer, (http://www.computerhistory.org/collections/catalog/102710223), pristup ostvaren 20.1.2018.
[21] Wikipedia, Anthony C. Hearn (https://en.wikipedia.org/wiki/Anthony_C._Hearn), pristup ostvaren 20.1.2018.
[22] Martin L. Griss, (https://www.ece.cmu.edu/directory/department/faculty/G/Martin_Griss_4154.html), pristup ostvaren 20.1.2018.
[23] Wikipedia, Gerald Jay Sussman, (https://en.wikipedia.org/wiki/Gerald_Jay_Sussman), pristup ostvaren 20.1.2018.
[24] M. Rouse, Smalltalk, (http://whatis.techtarget.com/definition/Smalltalk), pristup ostvaren 20.1.2018.
[25] Wikipedia, X3J13, (https://en.wikipedia.org/wiki/X3J13)
[26] Lisp, Harish Karnick, Članak: Resonance, ožujak 2014.
[27] Britannica, Alonzo Church, (https://www.britannica.com/biography/Alonzo-Church), pristup ostvaren 20.9.2017.
[28] AutoLISP Developer's Guide , Autodesk, 2012., (http://docs.autodesk.com/ACDMAC/2013/ENU/PDFs/acdmac_2013_autolisp_developers_guide.pdf), pristup ostvaren 20.9.2017.
[29] J. P. Sanders, AutoLisp, http://www.jefferypsanders.com/autolispintr_set.html pristup ostvaren 15.6.2017.

SAŽETAK

Ovaj diplomski rad se bavi problematikom korištenja LISP modula za CAD aplikacije. Opisani su povijest i razvoj CAD sustava , njegova implementacija, kreiranje baze podataka i njegova povezanost s CAM sustavom. Te također obuhvaća i uvod u tehničko crtanje. Dok LISP dio obuhvaća kratku povijest te prikazuje najvažnije naredbe za rad s AutoLISP-om. Za praktični dio projekta korišteni su *AutoCAD 2013, progeCAD* i *Notepad* . U *Notepadu* je kreirano 6 različitih LISP modula za rad s AutoCAD-om.

Ključne riječi: CAD, LISP, AutoCAD, progeCAD, AutoLISP, dizajn, tehničko crtanje

ABSTRACT

This graduate thesis deals with the issue of LISP module usage for the CAD applications. It describes history and development of the CAD system, it's implementation, creating database and it's connection with CAM system. It also includes the introduction to technical drawing. While LISP part includes short history and it shows the most important commands to use in AutoLISP. For the practical part of the project AutoCAD 2013, progeCAD and Notepad were used. In the practical part 6 different LISP modules were created in Notepad for the interaction with AutoCAD.

Keywords: CAD, LISP, AutoCAD, progeCAD, AutoLISP, design, technical drawing

ŽIVOTOPIS

Vedran Mrkonjić rođen 01.09.1992. godine u Osijeku. U Osijeku je završio osnovnu školu "Mladost". Godine 2007. upisuje 3. gimnaziju Osijek u Osijeku, smjer prirodoslovno-matematički koju je uspješno završio 2011. godine. Godine 2011. upisuje Sveučilišni preddiplomski studij Računarstva na Elektrotehničkom fakultetu u Osijeku, koji uspješno završio 2014. godine nakon čega je na istom fakultetu upisao Sveučilišni diplomski studij, smjer procesno računarstvo.