Tampereen ammattikorkeakoulu

AutoLISP-ohjelmointikieli tekni-sessä suunnittelussa

Henri Karhu

OPINNÄYTETYÖ
Joulukuu 2019

Sähkö- ja automaatiotekniikan koulutus
Automaatiotekniikka

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Sähkö- ja automaatiotekniikan koulutus
Automaatiotekniikka

KARHU, HENRI:
AutoLISP-ohjelmointikieli teknisessä suunnittelussa

Opinnäytetyö 49 sivua, joista liitteitä 2 sivua
Joulukuu 2019

Työn tavoitteena on tehostaa teknisen suunnittelun mekaanista osuutta ohjelmallisesti. Työssä tarkastellaan teknisen piirtämisen taustaa ja kehitystä aina nykypäivään saakka ja mietitään seuraavaa askelta teknisen piirtämisen kehityksessä. Lisäksi työssä pyritään digitalisoimaan viimeisiä analogisia osia teknisestä piirtämisestä.

Työssä käydään nousujohteisesti lävitse AutoLISP-kielen perusteet typistetysti. AutoLISP-kielen perustehtäväksi voidaan nähdä teknisen piirtämisen helpottaminen ja sujuvoittaminen. Työssä käydään lävitse AutoLISP-ohjelmoinnin aloittamiseen tarvittavia kriittisiä asetteluita, toimintoja ja ohjelmointialustoja. Edellisten ohella aiheina on myös tietoturvallisuuden osuus tietokoneavusteisessa suunnittelussa ohjelmakoodien osalta.

Teknisen suunnittelun ja dokumentoinnin määrä trendi on lisääntyvä. Yhä enemmän ja monimutkaisempia kokonaisuuksia suunnitellaan jatkuvasti. Tehokkaampi suunnittelu vaatii tulevaisuudessa yhdistelmän teknisestä tietämyksestä ja ohjelmointitaidoista.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree programme of Electrical and Automation Engineering
Automation Engineering

KARHU, HENRI:
AutoLISP Programming Language in Technical Design

Bachelor's thesis 49 pages, appendices 2 pages
December 2019

The objective of this thesis is to improve the mechanical part of technical designing via programming. In this thesis, the historical background and development of technical design is examined, and the next steps (in technical design) are considered.. In addition, the goal was to digitize the remaining analogic parts of technical designing.

The basics of the AutoLISP programming language are presented in this thesis. Alleviating and streamlining technical design can be seen as the main purpose of AutoLISP. Furthermore, the thesis deals with the critical settings, functions and the application programming interface needed to start AutoLISP programming. In addition to the abovementioned, a adiscussion is also procided on cybersecurity in the context of scripts used in CAD.

The demand for technical design and documentation is increasing. More and more complex entities are constantly being designed. More efficient design requires

---

SISÄLLYS

LYHENTEET JA TERMIT

CAD	Computer-aided design
AutoCAD	AutoDesk-yhtiön tuottama suunnitteluohjelmisto
LISP	List Processing
AutoLISP	AutoCAD-ohjelmistolle suunniteltu ohjelmointikieli
API	Application programming interface, tunnetaan myös ohjelmoinnin rajapintana
SFSP	Support File Search Paths, tuettu tiedostopolku
Entiteetti	Entiteeteillä viitataan suunnittelussa tehtyihin näkyviin ja näkymättömiin objekteihin ohjelman piirtoalueella.
Ohjelmakoodi	Tekstipohjainen lista, mikä koostuu useista komennoista. Tunnetaan myös skripteinä (script) englannin kielisessä materiaalissa.
String	Tekstimuotoinen data-arvo
Integer	Numeerinen kokonaisluvun data-arvo
Real	Numeerinen desimaaliluvut salliva data-arvo
Block	Lohko, minkä sisältämät erilaiset objektit ovat ynnätty yhden nimen alle. Käytetään usein toistuvien kokonaisuuksien suunnittelemiseen.

1 JOHDANTO

Hyvin toteutettujen asioiden tai projektien taustalla on aina huomattava määrä suunnitelmia ja dokumentteja. Historian aikana arkkitehdit tai insinöörit ovat piirtäneet suunnitelmia useimmiten paperille ja myöhemmin tietokoneavusteisesti. Tunnetuimpana esimerkkinä teknisestä piirtämisestä voidaan pitää Leonardo da Vincin piirtämiä keksintöjä liittyen lentämiseen ja sodankäyntiin.
Analogisesta käsin piirtämisestä siirryttiin kuitenkin osittain digitaaliseen piirtämiseen 1900-luvulla. Digitaalitekniikan kehitys 1900-luvun jälkimmäisellä puoliskolla levittäytyi nopeasti teknisen dokumentoinnin piiriin ja mahdollisti useita aikaisemmin mahdottomia tai vaikeasti toteutettavia toimintoja.

Verrattaessa kynällä ja viivoittimella tehtävää teknistä piirtämistä tietokoneavusteiseen suunnitteluun voidaan huomata edistysaskelia. Suurimpana digitaalitekniikan mahdollistamana asiana on piirrettyjen dokumenttien tarkkuus. Yhteen kuvaan on mahdollisuus sisällyttää erikokoisia objekteja ilman tarkkuuden kärsimistä, koska skaalautuvuus ei ole enää riippuvainen fyysiselle paperille kynällä piirretyistä objekteista. Aikaisemmin usean paperin tietojen yhdistäminen paperille onnistui käyttämällä läpinäkyviä papereita piirrettäessä. Saman järjestelmän eri osakokonaisuudet piirrettiin läpinäkyville papereille ja paperien ollessa päällekkäin saatiin eri tasot näkyville. Nykyään monimutkaisten kokonaisuuksien piirtäminen yhdelle dokumentille on huomattavasti helpompaa, koska käytettävissä olevien tasojen määrärajoitukset voivat olla tuhansissa, riippuen tietenkin käytettävästä ohjelmistosta.

Siirtyminen kynästä hiireen on käytännössä tapahtunut täysin. Edeltävää siirtymää voitaisiin kuvata osuvasti digitalisaatiolla. Samaa kehityssuuntaa voidaan yrittää soveltaa myös tietokoneavusteisen suunnittelemisen viimeiseen analogiseen osaan, hiireen. Hiirellä voidaan siirtää analogista liikettä tietokoneelle ja tarkoituksena on korvata ainakin osittain näitä toimintoja esimerkiksi näppäimistöllä tai ohjelmilla.

Työssä pyritään kehittämään teknistä suunnittelua ohjelmointikielen avulla. Ohjelmoidessa rajoitteina toimivat yleensä käytettävän ohjelmointikielen kykenevyys ja käyttäjän mielikuvitus. Kehitettävää tietokoneavusteisessa piirtämisessä löytyy yksittäisten kuvien kohdalta mittatilaustyönä tehtyjen funktioiden kohdalta, esimerkiksi piirustuksen raameihin tai rajoille tehtävä automaattinen rajaus näppäinyhdistelmällä. Suunnittelijan käyttäjäkokemusta voidaan parantaa tekemällä usein käytettäville toimenpiteille valmiita funktioita, mitä voidaan kutsua aina tarvittaessa.

Toinen kehitettävä asia tietokoneavusteisen suunnittelun saralla on automatisointi ja laajojen muutosten työstäminen. Aikaisempaan yksittäisten kuvien funktioihin verrattuna laajojen muutosten tekeminen piirustuksiin on huomattavasti enemmän sidottuna ohjelmointiin. Suunnittelu ja piirtäminen on mahdollista ilman ylimääräisiä ohjelmia, mutta laajojen tietokantojen muokkaamiseen tarvitaan jonkin asteen ohjelmointia tueksi. Aihetta rajataan yksittäisten funktioiden sijaan laajojen muutosten suorittamiseen tietokoneavusteisessa suunnittelussa. Aiheen rajauksen jälkeenkin joitain asioita joudutaan jättämään ulkopuolelle, kuten rekisterien käyttäminen tai ulkopuolisten ohjelmien, kuten Microsoftin Excel- ja Word-ohjelmistojen, yhdistäminen CAD-ohjelmistoihin.

2 AUTOLISP TEKNISESSÄ SUUNNITTELUSSA

2.1 Tekninen suunnittelu

Tekninen tietokoneavusteinen suunnittelu on suhteellisen helppoa ja kynnys aloittamiseen on matala. Usein käytetyt funktiot ovat helposti löydettävissä, kuten suorakulmat, ympyrät ja vastaavat. Uudetkin käyttäjät pääsevät piirtämiseen mukaan helposti. Pian käyttäjä alkaa hyödyntämään erilaisia toimintoja, kuten Rectangle-toimintoa ja Object Snap-toimintaa. Kokeneemmat tekniset piirtäjät käyttävät näppäimistöä hyödykseen huomattavasti enemmän kuin aloittelijat. Esimerkiksi ympyrän piirtämiseen ei välttämättä tarvita hiiren syötettä ollenkaan. Lyhenteillä ja numeroilla päästään huomattavasti nopeampaan ja tarkempaan lopputulokseen kuin hiirellä, sillä on helpompaa kirjoittaa komentoriville tarkat käskyt kuin käyttää hiirtä funktion valitsemiseen ja arvojen määrittämiseen.

Hiiri on ainoa jäljellä oleva analogisuutta jäljittelevä osa tietokoneavusteisessa teknisessä piirtämisessä. Tätä analogisuudesta johtuvaa epätarkkuutta on voitu paikata esimerkiksi Grid- ja Grid Snap-aputoiminnoilla. AutoCAD-ohjelmistoa on kuitenkin mahdollista käyttää tehokkaasti ilman hiirtä, esimerkiksi näppäimistöllä. Näppäimistön tehokkaasta käyttämisestä seuraavana askeleena on luontevasti kirjoittamisen vähentäminen. Onko esimerkiksi tarpeellista kirjoittaa toistuvasti samaa ja pitkää litaniaa? Voisiko tätä *automatisoida*?

2.2 AutoLISP -ohjelmointikieli

AutoLISP on AutoCAD:n muokkaamiseen ja edistyneeseen käyttöön suunniteltu ohjelmointikieli. AutoLISP nimensä mukaan perustuu alkuperäiseen LISP-kieleen, minkä juuret ovat 1950-luvulla. LISP on lyhenne sanoista List Processing, mikä tarkoittaa listamaista suoritustapaa. Alkuperäinen LISP on myös toiseksi vanhin korkean tason ohjelmointikieli.

AutoLISP julkaistiin vuonna 1986 ohjelmointirajapintana AutoCAD:n versiolle 2.1. LISP-ohjelmointikieli oli ensimmäinen AutoCAD:in lisätty API ja tarkoituksena oli

helpottaa suunnittelussa ilmenneitä ongelmia. Huomattavaa on, että AutoLISP on pääasiallisesti AutoCAD:in suunniteltu ja tarkoitettu (2, s. 1-2).

AutoLISP-ohjelmien tekeminen voi vaikuttaa aluksi vanhahkolta ja epäintuitiiviselta, koska pääosin ohjelmat tehdään kirjoittamalla komentoja tekstieditoriin ja viemällä ne suoraan AutoCAD:in. Nykyään hyvinkin yleiset ohjelmien sisäiset vianetsintään tarkoitetut työkalut, missä ohjelma näyttää virheet, eivät ole käytettävissä AutoLISP -ohjelmointikielessä. Käyttäjän täytyy itse etsiä mahdolliset virheet ohjelmasta lisäämällä esimerkiksi kriittisten kohtien arvojen tulostaminen itse ohjelmakoodiin. Rikkinäiset ohjelmakoodit useasti ilmoittavat komentojen virheistä ohjelmaa ajettaessa, mutta vianetsintä on käyttäjän harteilla.

3 KOODAAMINEN AUTOCADISSA

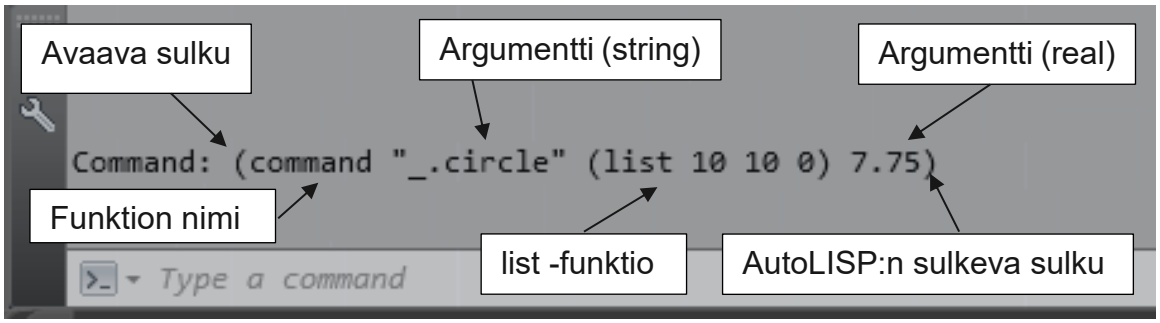
3.1 AutoLISP -syntaksi

AutoLISP-kieli on yhdistelmä tietokoneavusteista suunnittelua ja ohjelmointia. Tekniikan alalla useimmilla on kokemusta edes toisesta, mutta kokematonkaan ei ole millään tavalla estynyt opettelemaan AutoLISP-kieltä. Kynnys opettelemiseen on nimittäin erittäin matala ja tarvittavia ohjelmistoja on ainoastaan AutoCAD. AutoCAD sisältää nimittäin kaikki tarvittavat työkalut AutoLISP-ohjelmointiin. AutoCAD:n ja piirustuksen avaamisen jälkeen ohjelmoimaan pääsee kirjoittamalla yksittäisen avaavan sulkeen ja painamalla Enter-painiketta komentorivillä. Tästä ohjelma ymmärtää käyttäjän aloittavan ohjelmakoodin tai komennon kirjoittamisen. Samalla komentorivin syöte muuttuu kuvan 1 mukaiseksi ja käyttäjät tietää ohjelman siirtyneen AutoLISP:n suoritustilaan.

KUVA 1. AutoCAD komentorivin syöte valmiina ottamaan AutoLISP-käskyjä vastaan.

Takaisin piirtotilaan päästään painamalle ESC-näppäintä tai kirjoittamalla ensimmäiselle avaavalle sulkeelle pari. Useampien sulkujen avulla voidaan muodostaa sisäkkäisiä käskyjä. Sulkujen sijainnilla esimerkiksi eri riveillä ei ole väliä vaan ainoa vaatimus on vastapari jokaiselle sulkeelle. Sulkuparien parillisuuden tarkistaminen voi olla hankalaa laajemmissa ohjelmakoodeissa.

AutoLISP-kielen syntaksi koostuu argumenteista, komennoista, sulkeista muodostuvista kehysrakenteista ja atomeista (1, s. 17). Kuvassa 2 nähdään AutoLISP:n syntaksi ympyrän piirtämiselle. Komentoa seuraavia yksittäisiä data-arvoja kutsutaan atomeiksi. Esimerkiksi arvon edellä välittömästi oleva huutomerkki (!) lukee arvon ja heittomerkki (’) muodostaa atomeista listan eivätkä nämä yhdistelmät ole enää atomeita. Yksittäinen arvo mitä ei voida jakaa osiin on atomi. Listaan kuuluvat arvot eivät muodosta yhdessä atomia, mutta lista itsessään on jakamaton ja täten atomi. Argumentit voivat olla string, integer tai real muodossa.



KUVA 2. AutoLISP syntaksi.

Huomioitavaa on myös välilyöntien käyttäminen komentojen ja arvojen yhteydessä. Erilaisten loogisten tai matemaattisten operaattorien yhteydessä olevat arvot nähdään yhtenä komentona ilman välilyöntiä. Välilyönti tarkoittaa AutoLISP:lle komennon tai argumentin loppumista ja toisen alkamista.

3.2 AutoLISP-funktiot

Lähes kaikki AutoCAD:n toiminnat on mahdollista tehdä AutoLISP:n avulla. Lisäksi AutoLISP:lla voidaan suorittaa erilaisia aritmeettisia toimenpiteitä, muokata graafisia/symbolisia objekteja tai lisätä tiedostoihin ylimääräistä muistitilaa. Vaikka AutoLISP lisäisikin normaaliin AutoCAD piirtämiseen paljon uutta, niin suurin osa ohjelmakoodeista on tehty parantamaan käyttökokemusta funktioilla tai tekemään laajoja muutoksia useampaan piirustukseen.

Yksinkertaisena, mutta monipuolisena esimerkkinä AutoLISP:n käytöstä voidaan tehdä funktio kirjoittamalla rivit yksittäin komentoriville tai kopioimalla ja liittämällä ohjelmakoodi. Erilaiset komennot ja toiminnallisuus selitetään ohjelmakoodin jälkeisessä osassa.

```
(setq kierros 0)
(setq sivu1 20)
(setq sivu2 20)
(command ”rectangle” (list 5 5)’(sivu1 sivu2))
(repeat 10
(setq kierros (+ 1 kierros))
(setq sivu1 (* sivu1 1.4))
(setq sivu2 (* sivu2 1.4))
(command ”rectangle” ’(5 5) (list sivu1 sivu2))
(command ”change” (entlast) ”” ”p” ”c” kierros ””)
)
```

Ensimmäisellä kolmella rivillä setq-komento asettaa string-muotoisen argumentin globaaliksi muuttujaksi atomin määrittämään arvoon. Esimerkiksi globaalimuuttuja ”kierros” on setq-komennon muutettu arvoon 0. Huomioitavaa on globaalimuuttujia käytettäessä, että ne voivat olla jo muiden ohjelmakoodien käytössä tai niitä voidaan muuttaa muiden ohjelmakoodien toimesta.

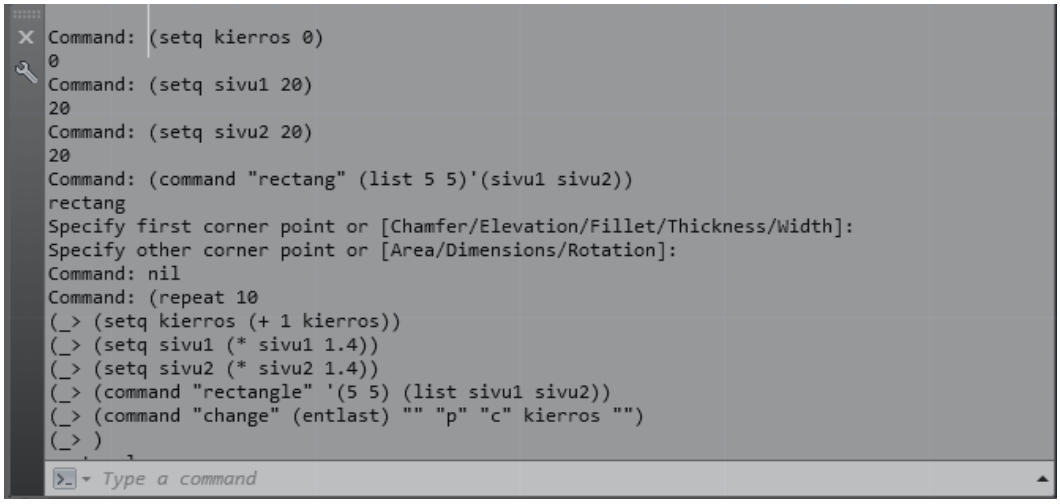
AutoCAD:ssa on paljon valmiita näppäinyhdistelmiä, mitä voidaan käyttää command-käskyllä. Commandin jälkeinen argumentti hakee ohjelmistossa olevia valmiita komentoja. Esimerkiksi ohjelmakoodissa oleva ”rectangle” piirtää suorakulmion käyttäjän syöttämän neljän arvon perusteella, kuten AutoCAD:n piirtotilassa käytettävä RECTANG-komento.

Silmukoiden tekemiseen on kaksi vaihtoehtoa; While ja Repeat. While-silmukkaa suoritetaan kunnes määritelty ehto täyttyy. Edeltävä toiminnallisuus sopii hyvin ohjelmakoodeihin, missä ei tiedetä muuttujien määrää tai laatua ennalta, joten toimintaa suoritetaan kunnes ehdot täytetään. Repeat-silmukka puolestaan toimii Integer-muuttujan perusteella. Silmukkaa toistetaan määrätyn positiivisen numeron perusteella. Staattisen luonteensa puolesta Repeat-silmukka sopii toimintoihin mihin ympäristön muuttujilla ei ole merkitystä.

Aritmeettisiä funktioita on useampi kymmentä millä voidaan manipuloida numeerisia int-, real- tai float-muuttujia. Kaikkien aritmeettisten funktioiden syntaksi on samantapainen. Esimerkkinä voidaan käyttää kuvan 3 jakamisen syntaksia. Ensimmäinen atomi (=numero) on jaettava ja jälkimmäiset ovat aina jakajia.

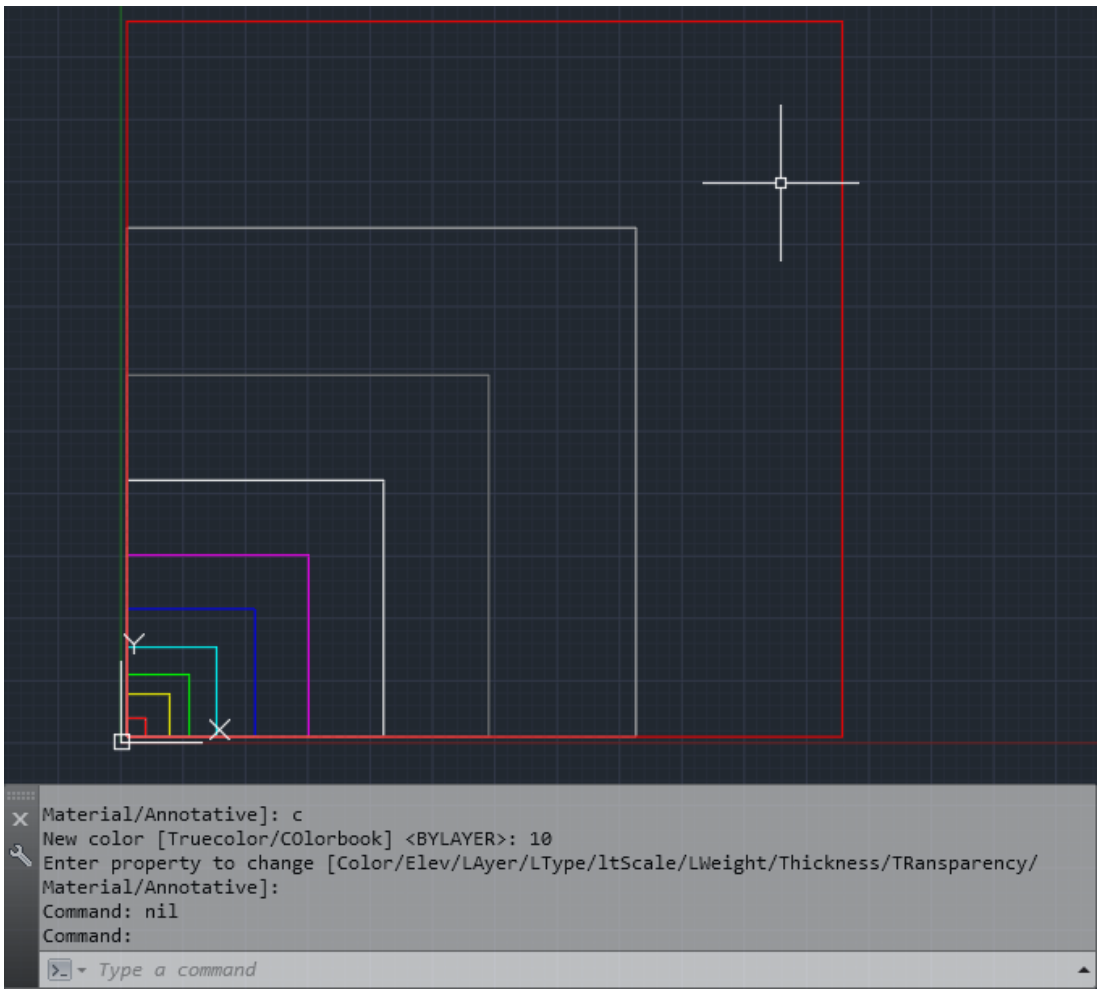
KUVA 3. AutoLISP:n jakamis-funktion toiminta.

Change-komennolla voidaan muokata valittua objektia erilaisilla tavoilla. Komennon käyttämiseksi kuitenkin vaaditaan objektin valitseminen piirustuksesta. Entlast on entiteettien muokkaamiseen liittyvä komento, mikä valitsee viimeisimmän piirustukseen lisätyn objektin. Objektin valinnan jälkeen change-komennolle annetaan käskyjä heittomerkkien avulla. Esimerkiksi pelkkien heittomerkkien käyttäminen (””) vastaa välilyöntiä. Kuvassa 4 on AutoCAD:iin kopioitu ohjelmakoodi.



KUVA 4. AutoCAD:in kopioitu ohjelmakoodi.

Ohjelmakoodin lopettavan sulkeen jälkeen tuloksena tulisi olla suorakulmio, mikä piirretään 1,4-kertaiseksi verrattuna edelliseen suorakulmioon jokaisella silmukan kierroksella. Ohjelmakoodin piirrettyä suorakulmion change-komento vaihtaa värit kierros-muuttujan mukaan, mikä perustuu suoritettujen silmukkojen määrään. Numeroarvo muuttuu väriksi riippuen ACI:n (AutoCAD Color Index) väripaletista. Lopputuloksen pitäisi näyttää kuvan 5 mukaiselta.



KUVA 5. AutoLISP -ohjelmakoodin lopputulos.

4 HALLINNOINTI

4.1 Yleistä

Käskyjen ja funktioiden syöttäminen AutoCAD:n komentoriville on hyvä tapa aloittaa AutoLISP –kielen kanssa. Jatkuvassa käytössä oleville komennoille voitaisiin joutua kuitenkin kirjoittamaan pitkiä tai monimutkaisia argumentteja, mikä taas olisi vastoin alkuperäistä ideaa piirtämisen helpottamisesta. Ainaisen kirjoittamisen sijaan on mahdollista säilöä näitä AutoLISP:n komentoja lsp-päätteisiin ASCII-tekstitiedostoihin. Näitä tekstitiedostoja voidaan lukea ja suorittaa AutoCAD:n komentorivillä. Komentojen lisäksi tekstitiedostoon on mahdollista lisätä suorittamatonta tekstiä (tunnetaan myös yleisesti kommentteina), esimerkiksi tekijä, päivämäärä tai selityksiä komentojen yhteyteen. Ennen LSP-tiedostojen ajamista AutoCAD:lla on hyvä suorittaa erilaisia asetuksia suunnitteluohjelmistoon. Ensimmäinen tarpeellinen asetus on tiedostojen sijainti ja käytettävien tiedostomuotojen turvallisuus, esimerkiksi yleisimmin käytetyt LSP-, DVB- ja ARX-tiedostomuodot saattavat sisältää vaarallisia ohjelmia sisällään. Jälkimmäinen tärkeä asetus on tiedostojen lataaminen AutoCAD:in. Lataaminen on mahdollista tehdä manuaalisesti, automaattisesti käynnistyksessä tai tarpeen mukaisesti.

AutoLISP –ohjelmakoodeja voidaan käyttää AutoCAD:ssa usealla erilaisella tavalla. Yksinkertaisin tapa on komentoriville kirjoittaminen, mutta tämä tapa on myös hitain eikä välttämättä suoraviivaisin. Parempana tapana voidaan pitää AutoLISP-tiedoston tuomista suoraan AutoCAD:in, esimerkiksi lsp-tiedoston tuominen tapahtuu klikkailemalla Manage -> Load Application -> tiedoston lataaminen. Kolmas tapa on komentomakron käyttäminen kustomoidun käyttäjäliittymän kautta (Customize User Interface). Näistä kolmesta ohjelmakoodien tuominen suoraan kansioista on suosituin ja yksinkertaisin tapa suorittaa komentoja.

4.2 Ohjelmat AutoLISP -tiedostojen tekemiseen

AutoLISP -ohjelmakoodien tekemisen vaatimuksena ei sinänsä ole ylimääräisiä ohjelmistoja. Kaikki ohjelmistot ovat käyttöjärjestelmästä riippuen joko integroitu AutoCAD:in tai tulevat käyttöjärjestelmän mukana. Kaikilla tavoilla voidaan tehdä

lsp- tai mnl-muotoisia tiedostoja (1, s. 272). Eroja löytyy ohjelmistojen väliltä, mutta noviisien on suositeltavaa käyttää ainakin aluksi yksinkertaisinta vaihtoehtoa eli Notepadia.

Notepad on kaikkien Windows –käyttöjärjestelmien mukana tuleva tekstieditori. Notepad:lla on mahdollista kirjoittaa ja muokata ASCII-muotoista tekstiä ja oletettu tallennusmuoto on txt, mikä joudutaan vaihtamaan. Notepad ei ole suunniteltu tai virallinen ohjelma AutoLISP:lle, mutta tiedostopäätteen muuttaminen lsp muotoon mahdollistaa tiedoston käyttämisen.

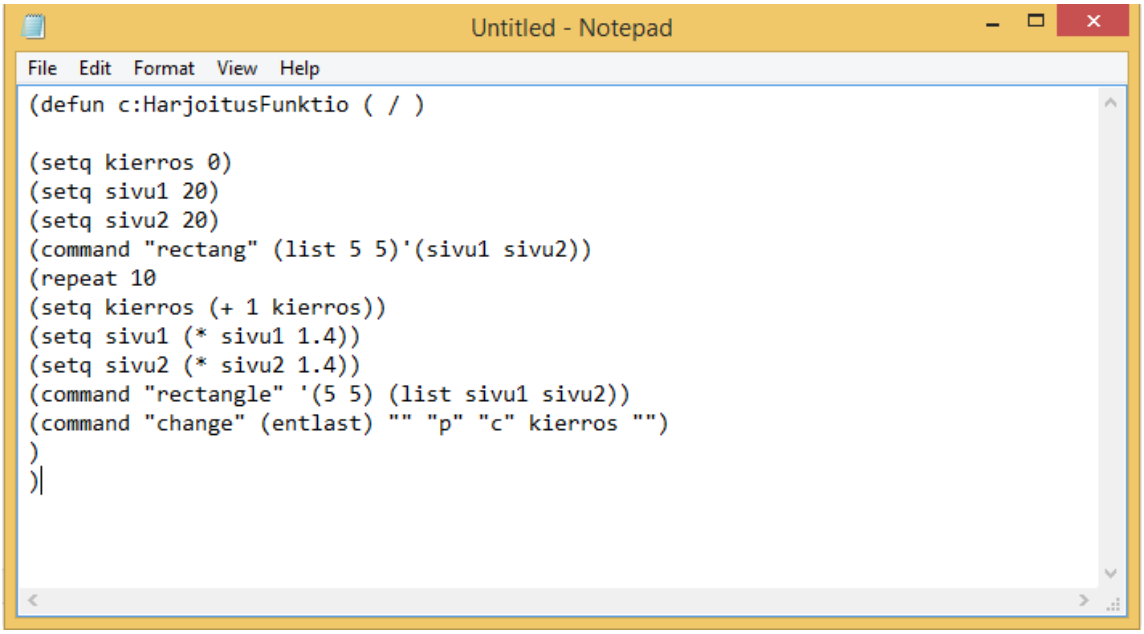
Visual LISP Integrated Development Environment tai tiiviimmin ilmaistuna **VLIDE** on AutoCAD:n sisäinen ohjelma, joka on tarkoitettu eritoten AutoLISP:lle. Kyseinen ohjelma käynnistetään AutoCAD:n komentorivillä aikaisemmin mainitulla lyhenteellä, VLIDE. Tekstieditori tunnistaa AutoLISP:n syntaksia ja värikoodaa tunnistetut komennot ja merkit omiksi väreikseen. Esimerkiksi sulkujen toinen pari on helpompaa löytää, sekä atomit ja argumentit on väritetty eri väreillä. VLIDE:en sisältyy myös vianetsintä työkalu, millä pystytään ajamaan ohjelmakoodia riveittäin virheen löytämiseksi.

TextEdit on Mac-käyttöjärjestelmän perus tekstieditori ja vastaa käytännössä Windows-käyttöjärjestelmän Notepad -ohjelmaa. Huomion arvoinen seikka on myös VLIDE:n puuttuminen Mac-käyttöjärjestelmistä, joten ainoaksi vaihtoehdoksi jää TextEdit:n käyttäminen.

4.3 AutoLISP-tiedoston tekeminen

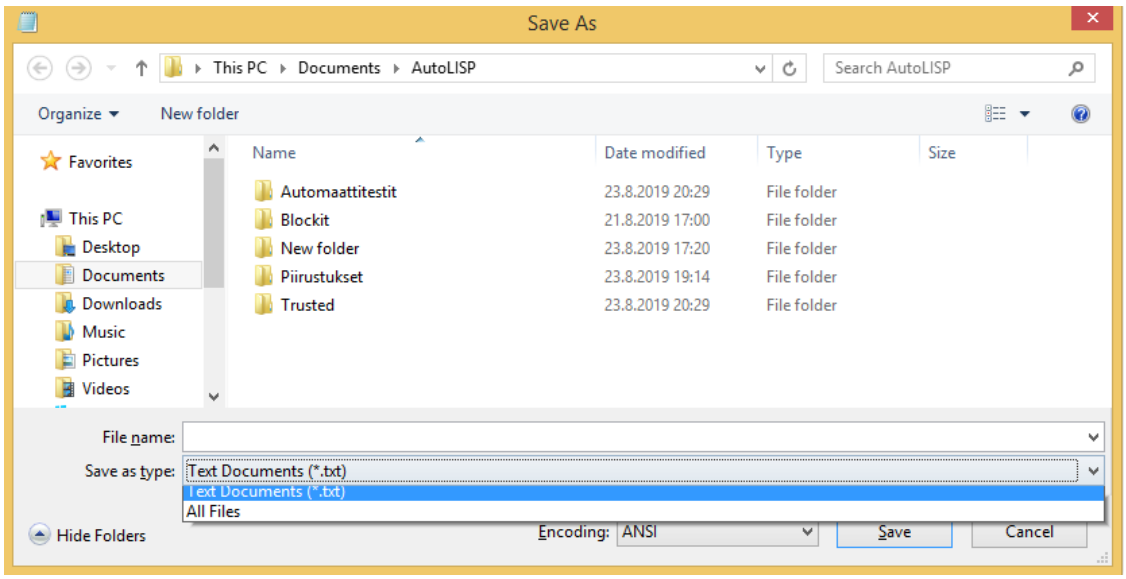
LSP-päätteiset tiedostot on yksi yleisimmistä tavoista tehdä ja säilöä AutoLISP-ohjelmakoodeja. LSP-päätettä käytetään pääosin piirtämiseen ja entiteettien muokkaamiseen liittyviin toimintoihin, vaikka ohjelmointikieli taipuisikin huomattavasti monimutkaisempiin tehtäviin. AutoCAD:ssa on myös useita muita tiedostomuotoja, mutta näitä käytetään hyvin rajatusti, kuten esimerkiksi mnl-pääte liittyy valikkoihin ja niiden muokkaamiseen.

Esimerkkinä LSP-tiedoston tekemisestä voidaan muuttaa aikaisemman 3.2 kohdan ohjelmakoodi LSP-tiedostoksi. Käytetään tekstieditorina Windows-käyttöjärjestelmän Notepad-ohjelmaa. Ohjelmakoodi kirjoitetaan tai kopioidaan Notepad:in ja tehdään pieniä lisäyksiä käytettävyyteen. Kuvassa 6 on lisätty alkuun ohjelmakoodin muuttaminen AutoCADissa kutsuttavaksi funktioksi. Ohjelmakoodi voidaan kutsua kirjoittamalla ”HarjoitusFunktio” komentoriville ja AutoLISP suorittaa käskyt.



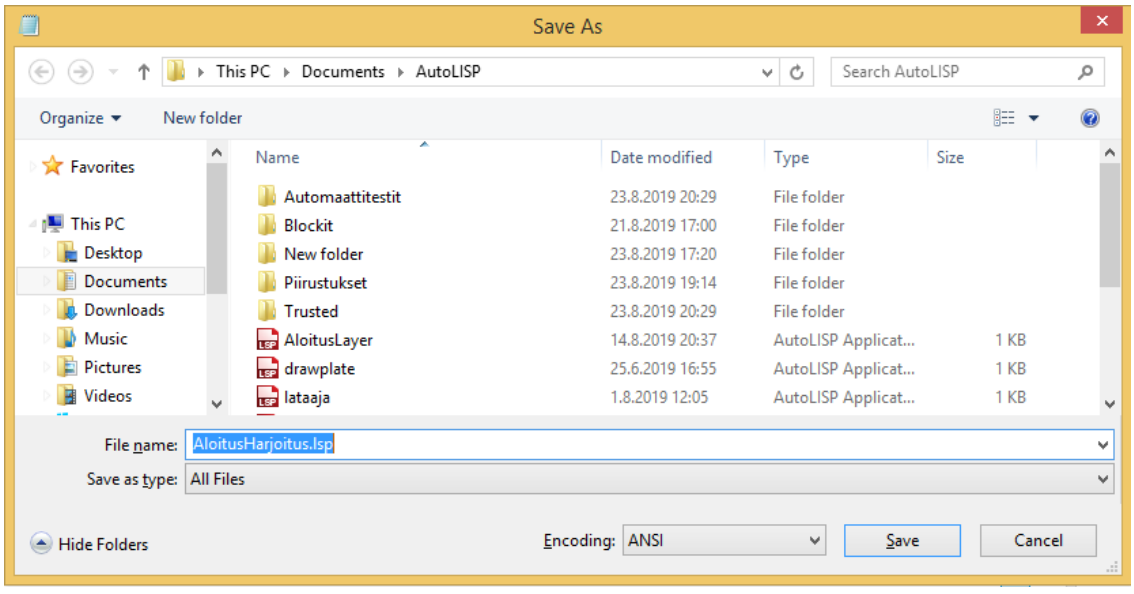
KUVA 6. Kohdan 3.2 AutoLISP–ohjelmakoodi muutettuna.

Valitaan ylävasemmalta File ja Save, valitaan haluttu kansio. Huomautuksena on suositeltavaa tehdä oma kansio tai hakemisto kaikille AutoCAD:in liittyville ohjelmakoodeille ja tiedostoille. Vaihdetaan kuvan 7 tavoin tiedostotyypiksi ”All Files”.



KUVA 7. Tiedostotyypin vaihtaminen.

Seuraavaksi tiedosto voidaan nimetä ohjelmakoodia mahdollisimman hyvin kuvailevaksi. Tärkeimpänä asiana on kuitenkin tiedostopäätteen muuttaminen manuaalisesti muotoon ”.lsp”. Tämä tapahtuu lisäämällä tiedoston loppuun .lsp. Verrattaessa kuvien 7 ja 8 tiedostoja voidaan nähdä AutoLISP-tiedostojen näkyminen ”All Files” valinnan jälkeen.



KUVA 8. Tallennettava tiedosto muutettuna ja muita .lsp –tiedostoja.

LSP-tiedosto ja sen sisältämä ohjelmakoodi on nyt jatkossa helposti ladattavissa ja käytettävissä AutoCAD-ohjelmistossa. Tiedostojen lataamisesta kerrotaan yksityiskohtaisemmin seuraavassa kappaleessa, mutta tiedosto voidaan ladata AutoCAD:in raahaamalla se piirustustilaan kansiosta.

4.4 Nimeämiskäytäntöjä

Ohjelmakoodit ja funktiot on kuitenkin käytännöllistä yhdistää johonkin kirjainyhdistelmään, mikä voidaan normaalin käytön yhteydessä nopeasti ottaa käyttöön. Esimerkkinä voidaan käyttää tarkennuksen tuomista kuvan objektien rajoille lyhenteellä ZE (=Zoom Entities). Lyhenteiden käyttöä rajoittaa kuitenkin AutoCAD:n omat käytössä olevat lyhenteet, joten ennen lopullista nimeämistä kannattaa tarkastaa vapaana olevat lyhenteet sivulta *https://www.autodesk.com/shortcuts/autocad*.

AutoLISP-funktioiden nimeämiseen ei kuitenkaan ole mitään virallista tai epävirallista käytäntöä, joten päällekkäisyydet funktioiden tai muuttujien nimeämisessä on mahdollisia. Esimerkiksi käytettäessä jonkun toisen AutoLISP-tiedostoja voi käytössä olla samoin nimettyjä funktioita tai muuttujia, mitkä eivät toimi halutulla tavalla. Yhtenä ratkaisuna on käyttää muuttujien nimeämisessä esimerkiksi omaa tai yrityksen nimeä. Esimerkiksi SQR- tai EntityList-funktioita kutsutaan nimellä karhu_SQR tai karhu_EntityList. Liittämällä henkilökohtaisen etuliitteen funktioihin ja muuttujiin riskit päällekkäisyyksille pienenevät huomattavasti.

4.5 Manuaalinen lataaminen

Kuten aikaisemmin mainittiin, ohjelmakoodien lataamiselle on kolme erilaista vaihtoehtoa; manuaalinen, automaattinen ja tarpeeseen perustuva. Riippuen valitusta tavasta ladata ohjelmakoodi on lisäksi erilaisia vaihtoehtoja sen toteuttamiselle. Ensimmäisenä käydään lävitse yleisin ja suositelluin tapa käyttää ohjelmakoodeja, nimittäin manuaalinen.
Manuaalisella lataamisella tarkoitetaan ohjelmakoodien lataamista AutoCAD:in käsin. Käyttäjä valitsee haluamansa ohjelmakoodit ja lataa ne käyttäen hyväksi erilaisia toimenpiteitä. Tämä on suositeltu tapa käyttää AutoLISP-tiedostoja, koska AutoCAD- ja muut suunnitteluohjelistot ovat usein suorituskykyä vaativia ja turhien ohjelmakoodien lataaminen pelkästään kuormittaa ohjelmistoa. Ohjelmakoodien lataamisen jälkeen AutoCAD pitää muistissaan jokaiselle piirustukselle ladattuja ohjelmakoodeja aina ohjelman tai piirustuksen sulkemiseen

saakka. Manuaaliseen ohjelmakoodien lataamiseen on useampia erilaisia tapoja (1, s. 281-282).

Appload –komento tai kuvakkeen painaminen avaa tiedosto-ikkunan. Etsittyäsi ja valittuasi AutoLISP kansiosi AutoCAD muistaa viimeksi käytetyn tiedostopolun. Valitsemalla tiedostoja ja painamalla Load-painiketta voit ladata valmiita AutoLISP-ohjelmakoodeja piirustuksiin.

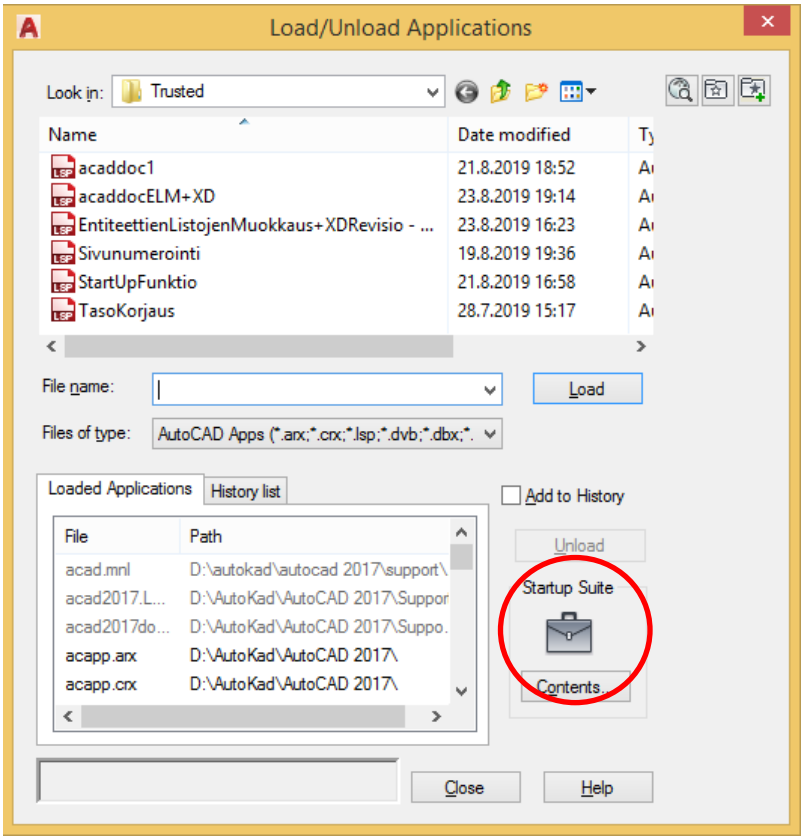
Raahaaminen suoraan kansiosta AutoCAD:in on myös mahdollista. AutoLISP -kansion avaaminen resurssienhallinnalla (File explorer) ja tiedostojen klikkaaminen ja raahaaminen AutoCAD:n piirustustilaan lataa ohjelmakoodit tälle piirustukselle.

Load –komento yhdistettynä ohjelmakoodin tiedostopolkuun lataa valitun ohjelmakoodin piirustukseen. Load–komentoa voidaan käyttää komentoriviltä, LSP-tiedostosta, CUI/CUIx-tiedostoista (Customize user interface) tai muun ohjelmakoodin kautta. Pysyvä tiedostopolku ei ole aina pakollinen komennon käyttämiseen, vaan on mahdollista käyttää SFSP-ominaisuutta (Support File Search Path). SFSP-ominaisuuden käyttäminen vaatii kuitenkin oletusarvoista poikkeavia asetuksia, mitkä käydään lävitse myöhemmässä vaiheessa.

4.6 Automaattinen lataaminen

Automaattisesti ladattaville AutoLISP-tiedostoille pyritään useimmiten löytämään dokumenteista joitakin yhtäläisyyksiä. Esimerkiksi useasti käytettävät funktiot tai kaikkiin kuviin tulevat asettelut ovat suositeltuja automaattisesti ladattaviksi tiedostoiksi. Liian erityiset tai harvoin käytettävät funktiot ainoastaan kuormittavat suorituskykyä käyttämättöminä ja ovat täten suositeltuja manuaalisesti ladattaviksi. Automaattiseen ohjelmakoodien lataamiseen on useampi erilainen tapa, kuten manuaalisessa lataamisessa (1, s. 282-283).

Startup Suite – (APPLOAD–komento) aukaisee tutun näkymän ohjelmakoodien lataamiselle, mutta automaattisen lataamisen saavuttamiseksi tiedostot lisätään *Startup Suite*-nimiseen kansioon. Nimensä mukaisesti Startup Suite lataa kaikki sisältämänsä ohjelmakoodit piirustukseen avaamisen jälkeen. Tiedostojen poistaminen tästä kansiosta ei poista nykyisissä piirustuksissa olevia tiedostoja, vaan tiedostojen poistaminen tehdään sulkemalla piirustus tai käyttämällä unload-komentoa. Kuvassa 9 nähdään Startup Suite Apload –komennon käyttämisen jälkeen.



KUVA 9. Startup Suite.

Nimeämällä tiedoston/tiedostot tarkasti valikoiduilla nimillä mahdollistaa lataamisen automaattisesti AutoCAD:lla. AutoCAD lataa oletusarvoisesti tietyn nimiset tiedostot sisällöstä huolimatta. Aloituksessa ladattavien tiedostojen nimet on taulukossa 1 prioriteettien mukaisessa järjestyksessä, ylhäällä korkeimman prioriteetin tiedosto (3, s. 985-988). Näistä on suositeltavaa käyttää omille ohjelmille erityisesti acad.lsp-, acaddoc.lsp- ta acad.mnl –tiedostoja.

TAULUKKO 1. Automaattisesti latautuvat tiedostot.

acad.rx	Lataa ObjectARX-tiedostoja.
acad2017.lsp	AutoCAD:n AutoLISP-tiedostot. Tiedostot ovat tehty asennusten yhteydessä ja ovat vapaasti muokattavissa. Versio riippuvainen.
acad.lsp	Tiedosto ladataan uutta piirustusta avattaessa, jos acadlspasdoc-muuttujan arvo on 1. Tiedosto on tehtävä erikseen.
acad2017doc.lsp	AutoCAD:n AutoLISP-tiedostot. Tiedostot ovat tehty asennusten yhteydessä ja ovat vapaasti muokattavissa. Versio riippuvainen.
acaddoc.lsp	Tiedosto ladataan uutta piirustusta avattaessa. Tiedosto on tehtävä erikseen.
acad.mnl	AutoCAD:n multiline library-tiedosto.
tiedostonimi.mnl	AutoCAD:n käyttäjänäkymään liittyvien tiedostojen lataaminen.

Autoload –funktio mahdollistaa valittujen AutoLISP-tiedostojen yhdistämisen toisiin käyttäjän tekemiin funktioihin. Tällä voidaan tehdä yksittäinen tiedosto, mikä lataa muita funktioita piirustukseen. Toimii hyvin esimerkiksi s::startup –funktion kanssa.

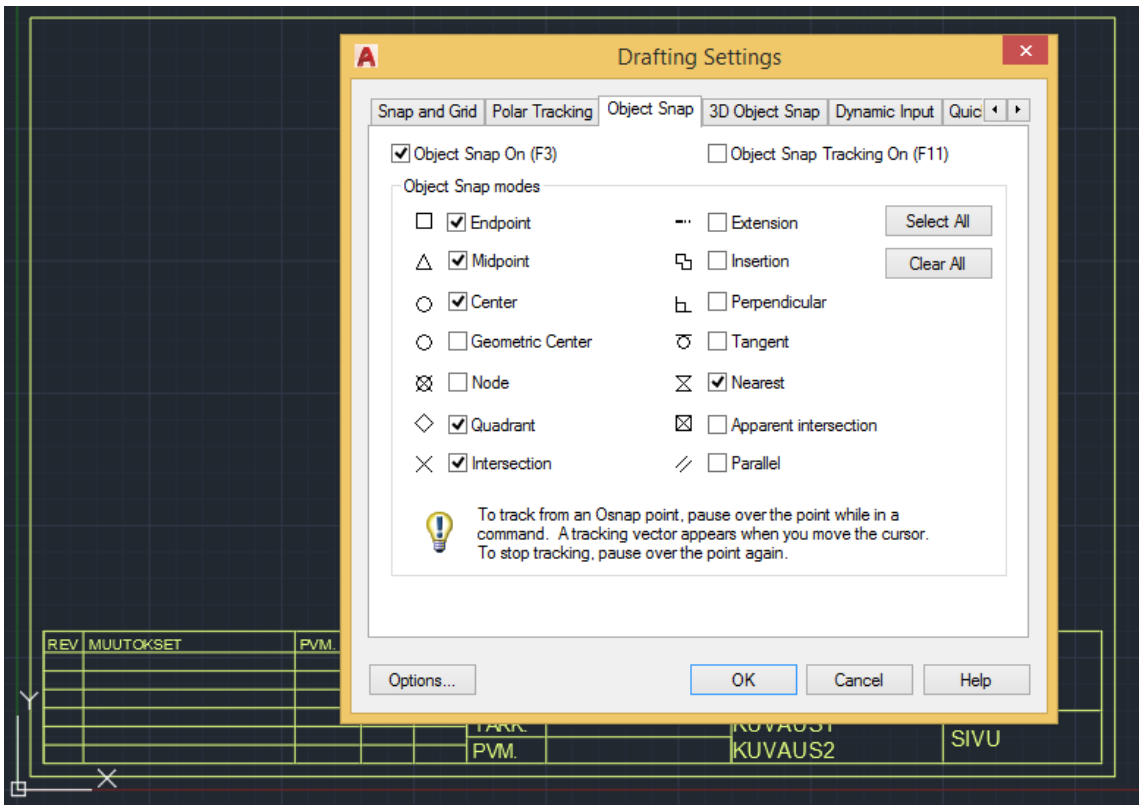
4.7 Aloitukseen liittyvät funktiot

AutoLISP:lla on mahdollista tehdä ainutlaatuinen funktio nimeltänsä *s::startup* (3, s. 1103). Funktion tarkoituksen voi päätellä nimestäkin, sillä funktio toteutetaan kun piirustus aukaistaan tai luodaan. Ainoana ehtona tämän suorittamiselle on piirustukseen ladattava lsp-tiedosto. Ainutlaatuiseksi *s::startup* –funktion tekee sen rajoitettavuus. Useamman lsp-tiedoston sisältäessä *s::startup*-funktion ainoastaan viimeisin ladattu tiedosto pystyy suorittamaan ohjelmakoodinsa.

Startup-funktioon kannattaa sisällyttää ainoastaan hyvin yleisiä ja usein käytettyjä asioita. Jokaisella on omat mieltymyksensä, kun puhutaan suunnittelun asetuksista. Esimerkiksi optimaalisen tartunta-asetusten (Object Snap mode) valinta on täysin subjektiivista, joten ohjelmakoodista voi muokata ”osmode”-muuttujan haluamaansa tilaan. Muita asetuksia on ”pickfirst” ja ”attreq” asetukset. Pickfirst-asetuksella määritellään voiko objekteja valita ennen kuin mitään komentoa on annettu. Attreq-asetus määrittää blockeja piirustukseen tuotaessa, että kysytäänkö käyttäjältä blockin attribuuteille arvoja vai ei. Ohjelmakoodi tekee myös tason valmiiksi tuotavalle piirustuspohja-blockille. Tämän jälkeen piirustuksen suurennus tuodaan reunoja myötäileväksi. Ohjelmakoodi nähdään kokonaisuudessaan alapuolella.

```
(defun s::startup ( / Pohjustus)
                (setvar "osmode" 567)
                (setvar "pickfirst" 1)
                (command "_layer" "m" "Pohja" "c" "61" " " "")
                (setvar "attreq" 0)
                (command "-insert" "piirustuspohja_hka" "0,0" "1" "1" "0")
                (command "zoom" "e")
)
```

Nimeämällä yllä olevan ohjelman taulukon 1 mukaisesti acaddoc.lsp –nimiseksi AutoCAD avaa tiedoston automaattisesti ja s::startup suorittaa ohjelmakoodin välittömästi. Kuvassa 10 nähdään tulos avattaessa uusi piirustus AutoCAD:ssa.



KUVA 10. Aloitusfunktion tuoma piirtopohja taustalla ja tartunta-asetukset.

Ohjelmakoodissa oleva ””osmode” 567” viittaa valittuihin tartunta-asetuksiin. Taulukossa 2 on kaikki mahdolliset tartunta-asetukset. Useita asetuksia voidaan yhdistellä toisiinsa lisäämällä luvut keskenään. Esimerkiksi osmode 567 on summa NEA-, INT-, QUA-, CEN-, MID- ja END-muuttujista (6). Taulukossa 2 on kaikki mahdolliset tartunta-asetukset ja niiden arvot.

$$512 + 32 + 16 + 4 + 2 + 1 = 567$$

TAULUKKO 2. Taulukko tartunta-asetuksista.

0	NONe
1	ENDpoint
2	MIDpoint
4	CENter
8	NODe
16	QUAdrant
32	INTersection
64	INSertion
128	PERpendicular
256	TANgent
512	NEArest
1024	Geometric CEnter
2048	APParent Intersection
4096	EXTension
8192	PARallel
16384	Suppresses the current running object snaps

4.8 AutoLISP-tiedostojen sijainnit

AutoCAD ei oletusarvoisesti tiedä tehtyjen tiedostojen sijaintia tai muistialueita. Käyttäjä voi mielivaltaisesti valita LSP-tiedostoille halutun sijainnin tietokoneella. Yksin toimittaessa sijainti ei nouse oleelliseen asemaan, mutta esimerkiksi yrityksessä toimiessa ideaaliin sijaintiin vaikuttaa muutamat asiat. AutoLISP-tiedostojen tulisi olla yleisessä kansiossa kuvaavasti nimettynä, jotta tarvittavat ohjelmakoodit olisivat myös muiden työntekijöiden käytettävissä. Edellinen vaatii kuitenkin tiedostojen olevan ainoastaan luettavissa kaikilta paitsi määrätyiltä tiedostojen tekijöiltä tai päivittäjiltä.

Toinen turvallisuuteen liittyvä tekijä on tiedostojen luotettavuus ja oikeudet. Oletusarvona AutoCAD kysyy aina ennen AutoLISP-tiedoston lataamista, onko tiedosto luotettava ja sallitaanko lataaminen AutoCAD:in. Tämä ominaisuus on hyvä muistutus tuntemattomien AutoLISP-tiedostojen lataamisen vaaroista. Nimittäin vuonna 2012 havaittiin ensimmäiset AutoLISP-kielellä tehdyt virukset. Esimerkiksi ACAD/Medre.A oli virus, mikä levisi sähköpostin liitteiden kautta (4). Sähköpostin kautta lähetetyissä saastuneissa piirustuksissa oli AutoLISP-tiedosto, mikä kopioi itsensä mahdollisimman moneen tiedostosijaintiin ja pyrki käynnistämään itsensä uusien piirustusten mukana. Tämän lisäksi virus lähetti kaikki saastuttamansa piirustukset sähköpostin kautta kolmannelle osapuolelle. Piirustusten lisäksi virus keräsi rekisteristä kaikki henkilökohtaiset Office-ohjelmistoihin liittyvät tiedot. Edellinen esimerkki liittyi hienostuneeseen teollisuusvakoiluun, mutta on olemassa alkeellisempia ja lievästi haitallisiksi suunniteltuja viruksia mitkä esimerkiksi räjäyttävät kaikki objektit kuvasta (explode/burst –komennot). Haavoittuvuudet on korjattu vuoden AutoCAD 2013 version SP1-päivityksellä (Service Pack 1), mikä monimutkaisti tiedostojen automaattista lataamista ja suorittamista verrattuna aikaisempaan ohjelmaversioon (7).

Vuoden 2014 AutoCAD versiossa lisättiin ominaisuus nimeltä ”Trusted Locations” (1, s. 287-291). Ladattaessa AutoLISP-tiedostoja AutoCAD:in, AutoCAD varmistaa käyttäjältä luvan lataamiseen. Määrittelemällä tietty tiedostosijainti luotetuksi voidaan käyttäjän puolelta vaadittu varmistus ohittaa ja ladata AutoLISP-tiedostot suoraan piirustukseen.

1. Kirjoitetaan **Options** AutoCAD:n komentoriville.

1.B. Painetaan vasemmassa ylänurkassa olevaa AutoCAD-ikonia ja valitaan **Options**.

2. Valitaan välilehdistä **Files**.

3. Laajennetaan/Avataan **Trusted Locations**.

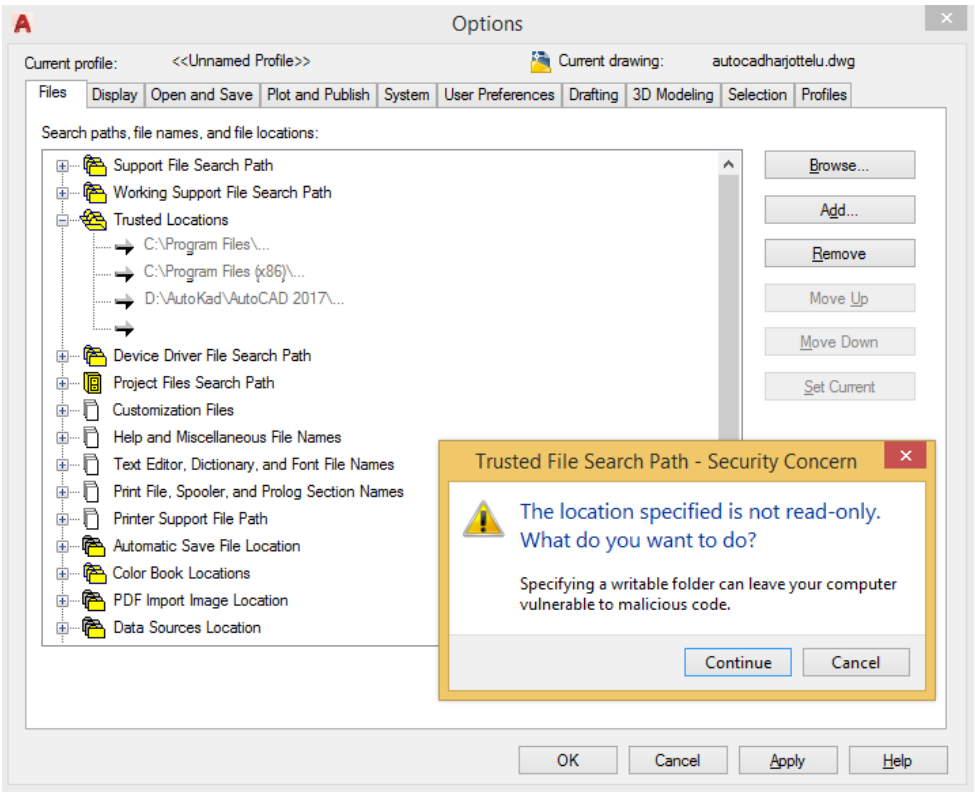
4.A. Oikealla puolella olevista painikkeista **Add**-painikkeella lisätään haluttu kansio luotetuksi kansioksi.

4.B. **Browse**-painikkeella voidaan avata resurssienhallinta kansion löytämisen helpottamiseksi.

4.C. **Remove**-painikkeella voidaan myös poistaa luotettuja kansioita.

5. Hyväksy kansio Trusted Files Search Path –kelpoiseksi, kuva 11.

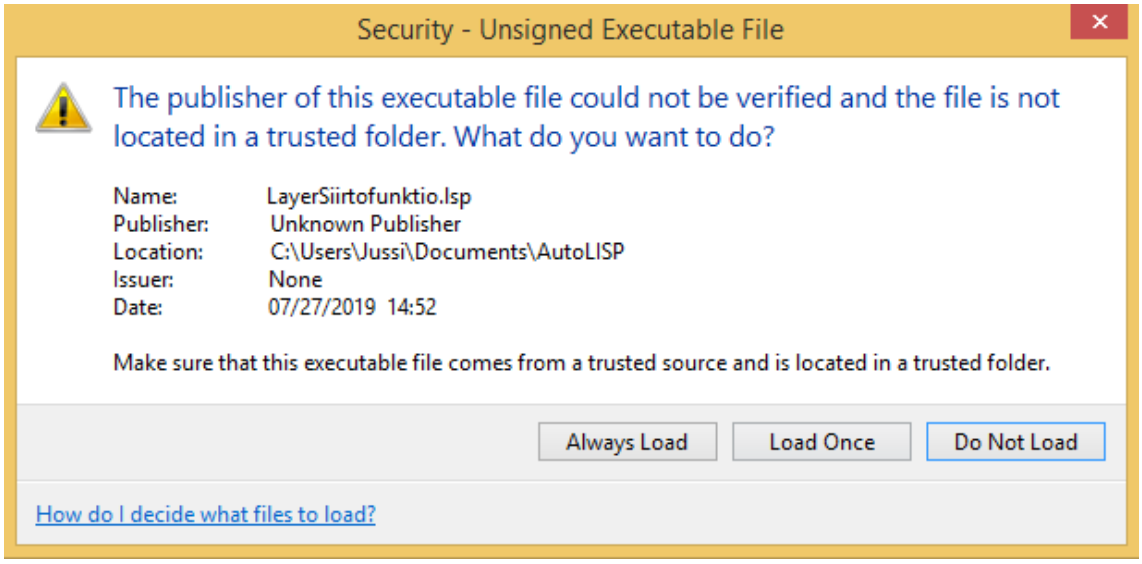
6. Paina OK hyväksyäksesi muutokset.



KUVA 11. Kuva asetuksista ja luotetun kansion hyväksymisestä.

Luotetun kansion luomisen jälkeen AutoCAD ei enää kysy varmistusta kansiosta ladatuista tiedostoista. Tämä pätee kaikkiin manuaalisiin ja automaattisiin latauksiin. Vuoden 2014 ja myöhemmissä AutoCAD versioissa on secureload –komento, millä voidaan määritellä turvallisuustaso ladatuille tiedostoille ja tiedosto-

sijainneille (8). Oletusarvona secureload arvona on 1, mikä sallii luotetuista kansioista lataamisen ilman käyttäjältä varmistamista ja lataamisen varmistamisen jälkeen muista sijainneista. Muita secureload arvoja on 0, mikä sallii kaikkien AutoLISP-tiedostojen lataamisen ilman käyttäjän varmistamista. Tätä turvallisuustasoa ei suositella käytettäväksi, mutta tällä vaihtoehdolla voidaan ajaa tietoturvallisuuspäivitystä edeltäviä ohjelmia. Secureload 2 sallii AutoLISP-tiedostojen lataamisen ainoastaan luotetuista kansioista. Kuvassa 12 on tiedoston lataaminen määrittelemättömästä kansiosta secureload arvon ollessa 1.



KUVA 12. Varmistus ennen ohjelman lataamista AutoCAD:in.

Luotettujen kansioiden tekemisen jälkeen AutoCAD:lle on hyvä tehdä hyvin samankaltainen toimenpide lataamisen sujuvuuteen liittyen. Resurssienhallintaan ja tiedostojen sijaintiin liittyvät AutoCAD-komennot vaativat käyttäjältä tiedostopolkujen määrittämistä erikseen jokaiselle käyttäjän tekemälle kansiolle. AutoCAD:ssa on valmiiksi määritelty pieni määrä Support File Search Path-nimisiä kansioita mistä ohjelma osaa etsiä tiedostoja, mutta nämä kansiot liittyvät ohjelmiston toimintaan. SFSP-tuen lisääminen piirustus- tai AutoLISP-kansioille tapahtuu samalla tavalla kuin luotetuille kansioille, kuvassa 13 näkyy SFSP-tuetut kansiot.

1.A. Kirjoitetaan **Options** AutoCAD:n komentoriville.

1.B. Painetaan vasemmassa ylänurkassa olevaa AutoCAD-ikonia ja valitaan **Options**.

2. Valitaan välilehdistä **Files**.

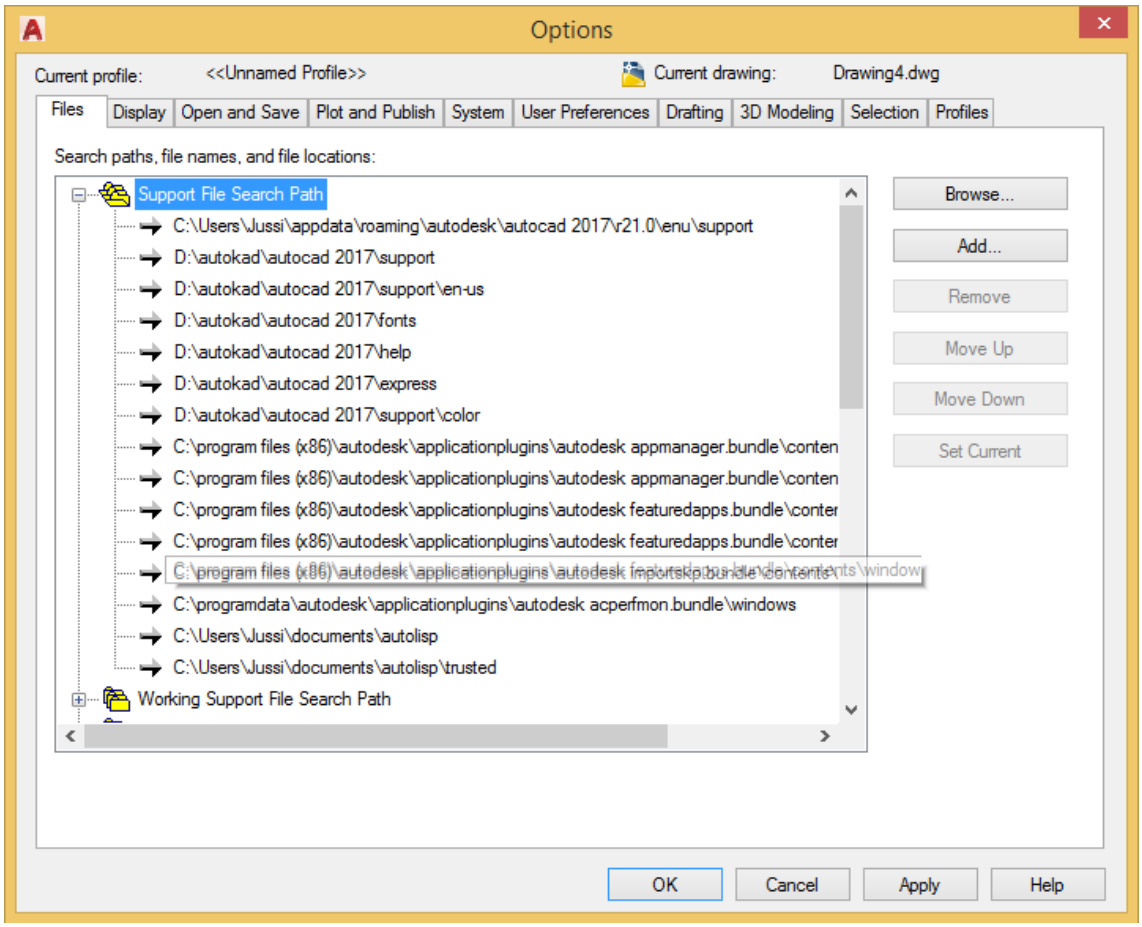
3. Laajennetaan/Avataan **Support File Search Path**.

4.A. Oikealla puolella olevista painikkeista **Add**-painikkeella lisätään tiedostosijainti AutoCAD:lle.

4.B. **Browse**-painikkeella voidaan avata resurssienhallinta kansion löytämisen helpottamiseksi.

4.C. **Remove**-painikkeella voidaan myös poistaa luotettuja kansioita.

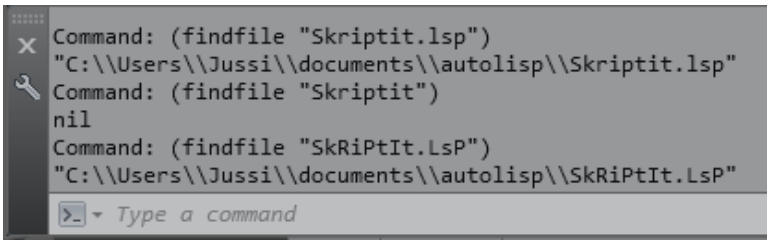
5. Paina OK hyväksyäksesi muutokset.



KUVA 13. SFSP-tuettujen kansioiden asetukset ja sijainnit.

Kansion ollessa SFSP-tuettu voidaan kansion sisältöä käsitellä komennoilla. Findfile-komennolla voidaan etsiä tiedostoja tiedostonimen ja –päätteen perusteella. Load-komennolla voidaan ladata tuetuista tiedostosijainneista haluttuja tiedostoja, kuten lsp- tai mnl-ohjelmakoodeja. Support File Search Path ja Trusted

Locations ovat kaksi erillistä asetusta vaikka ne vaikuttavatkin hyvin päällekkäisiltä. Yksinkertaistettuna Trusted Locations on ohjelmakoodeihin liittyvää sallimista ja Support File Search Path on AutoCAD:n asiasanoitukseen liittyvää tietoa. Joihinkin komentoihin tarvitaan kumpaakin ominaisuutta, kuten AutoLISP-tiedoston käynnistäessä toisen ohjelmakoodin suunnitteluohjelmiston ulkopuolisesta kansiosta. Molempien asetusten käyttäminen on suositeltua kansioille missä valmiit ja testatut AutoLISP-tiedostot sijaitsevat. Esimerkiksi Findfile-komennolla AutoCAD käy kaikki SFSP-määritellyt kansiot lävitse ja etsii lainausmerkein kohdennettua argumenttia. Haettavan tiedoston versaalilla ei ole väliä, mutta itse tekstin pitää olla ehdottoman tarkka tiedostopäätettä myöten, muutoin hakutulos on todennäköisesti nil. Kuvasta 14 nähdään komento ja palautetut arvot.



KUVA 14. Tiedoston etsiminen SFSP-tuetuista kansioista.

5 ENTITEETIT

5.1 Entiteeteistä yleisesti

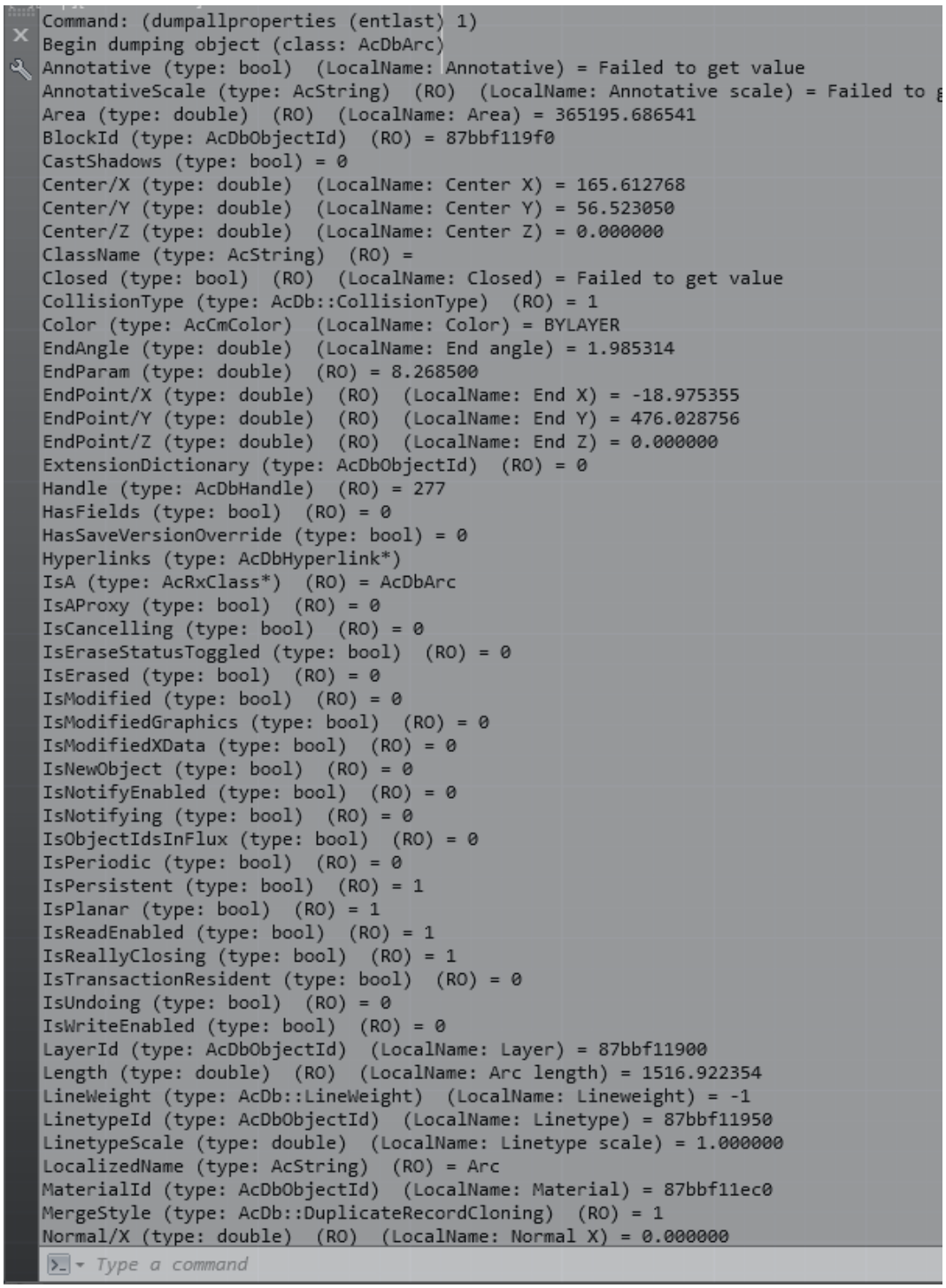
AutoLISP antaa käyttäjälle poikkeuksellisen laajat keinot muuttaa piirustuksen tietoja. Osoituksena tästä on tavanomaisilta käyttäjiltä piilotettuna olevia parametreja ja asetuksia. AutoCAD:n piirustuksissa muuteltavat tiedot voidaan jakaa yleisesti kahteen eri ryhmään; graafisiin ja epägraafisiin. Graafisilla entiteeteillä viitataan piirustuksessa de facto nähtäviin objekteihin. Nämä objektit ovat tutumpia käyttäjille ja muutamina näytteinä mainittakoon seuraavat (5, s. 421-422).

- LINES
- POLYLINES
- DIMENSIONS
- SOLIDS
- ARCS
- CIRCLES
- POINTS
- BLOCK INSERTS
- TEXT
- ATTRIBUTES
- MTEXT

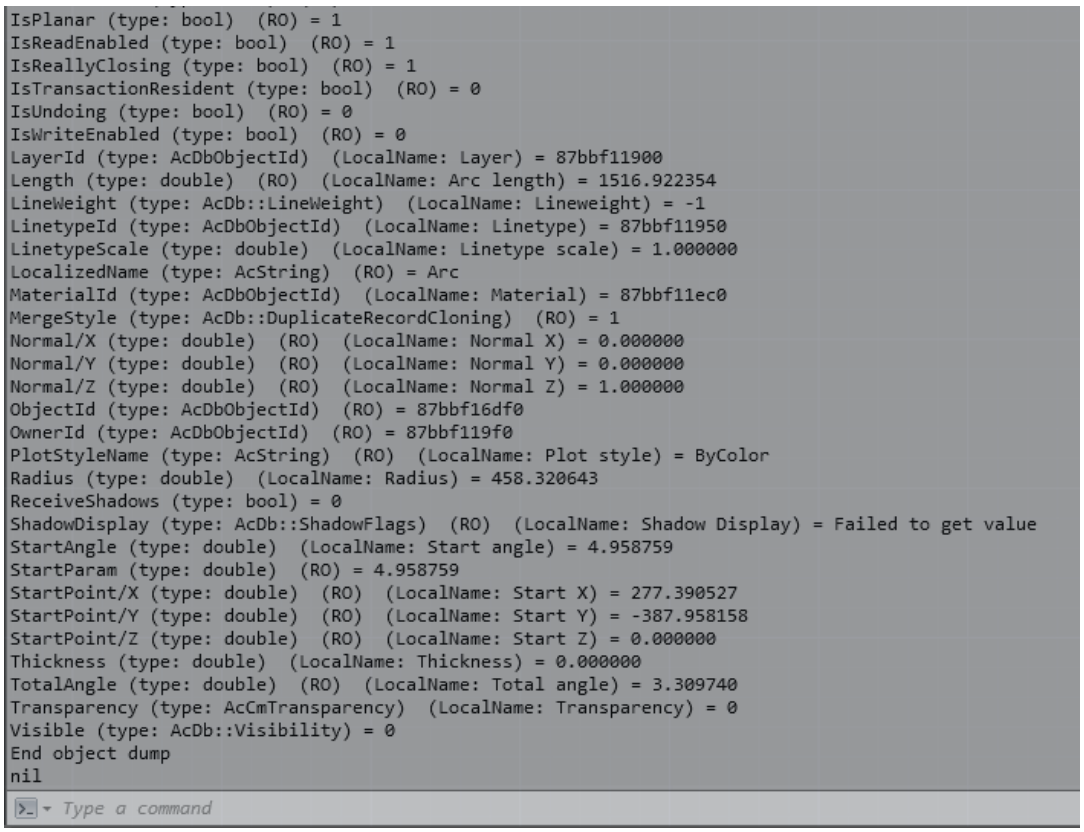
Epägraafiset entiteetit ovat abstraktimpia luonteeltansa ja liittyvät graafisiin objekteihin lähinnä epäsuorasti. Esimerkiksi läpinäkyvyydet, tasot tai tekstityylit eivät ole graafisia entiteettejä, mutta niillä voidaan vaikuttaa objektin visuaaliseen olemukseen välillisesti. Epägraafiset entiteetit voidaan jakaa kahteen pääryhmään; taulukkoihin (=symbol tables) ja sanastoihin (=dictionaries). Alapuolella on lista joistakin taulukko-muotoisista entiteeteistä (5, s. 484).

- APPID
- BLOCK
- DIMSTYLE
- LAYER
- STYLE
- UCS
- VIEW
- VPORT

Objekteja tai toiselta nimeltään entiteettejä voidaan tehdä ja muokata kahdella fundamentaalisesti erilaisella tavalla. Ensimmäinen on jo aikaisemmin näytettyjen komentojen avulla. Komennoilla pystytään tekemään kaikkein tavanomaisimmat asiat yksinkertaisesti ja nopeasti. Komentojen avulla tehtävät muutokset ovat kuitenkin rajoitettuja, koska AutoCADin kehittäjät päättävät tarpeellisista toiminnoista ja itse komentojen sisällä olevan ohjelmallisuuden. Tästä syystä ulkopuolelle voi jäädä joitakin ominaisuuksia, mitä voidaan tarvita erikoistapauksissa. Dumpallproperties-komennolla AutoLISP kirjoittaa kaikki objektiin liittyvät ominaisuudet. Otetaan näytteeksi juuri piirretyn kaaren ominaisuudet dumpallproperties-komennon ja entlast-funktion avulla, kuvissa 15 ja 16 tulokset.

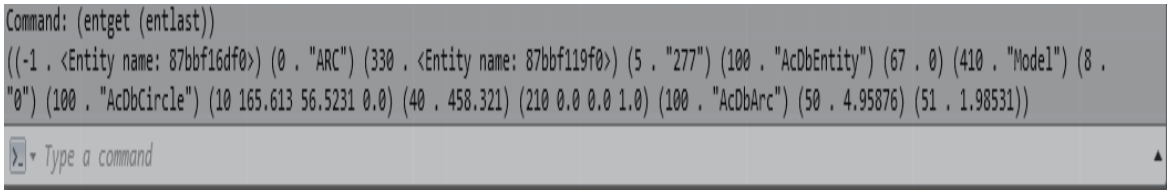


KUVA 15. Piirretyn kaaren ominaisuudet osa 1.



KUVA 16. Piirretyn kaaren ominaisuudet osa 2.

Toinen tapa manipuloida entiteettejä on erilaiset entiteetteihin liittyvät funktiot, joissa on useimmiten ent-etuliite. Funktioiden avulla tehtävät muokkaamiset ovat vanhempi tapa verrattuna komentoihin, mistä johtuen funktiot toimivat myös vanhemmissa versioissa kuin vuoden 2012 AutoCAD:ssa. EntGet-funktioilla (9) voidaan hakea objektin ominaisuudet samalla tavalla kuin dumpallproperties-komennon avulla. Kuvassa 17 nähdään funktion hakemat tiedot.



KUVA 17. Objektin ominaisuudet EntGet-funktiolla.

5.2 DXF-ryhmät

Verrattaessa dumpallproperties-komennon ja entget-funktiolla saatavia tietoja huomataan edellisen olevan huomattavasti laajempi ja kattavampi. Ero johtuu lähinnä esitettävästä tiedosta ja esitystavasta. Entiteettien avulla muokkaaminen ja luominen ovat tiedostotyypistä johtuen huomattavan tekstipohjaista. Selitykset tai lyhenteet erilaisille asetuksille tai määrittelyille ovat numeraalisessa muodossa minimaalisen datan käytön takia, koska pelkkien DXF-tiedostojen perusteella on mahdollista piirtää ja säilöä dokumentteja. DXF-tiedostot korvaantuvat kuitenkin jatkuvasti uudemmalla DWG-tiedostomuodolla, mutta DXF-muotoinen muokkaaminen on silti relevanttia AutoLISP:n osalta. Aikaisemmassa kuvassa 17 nähdään sulkujen sisällä aina ensimmäisenä numeroarvo. Tämä numeroarvo on DXF-ryhmäkoodi, mikä määrittää objektin ominaisuuden sulkeiden jälkimmäisen puoliskon datalla. Esimerkiksi DXF-ryhmäkoodi 10 määrittää aloituspisteen, keskipisteen, korkeuden (Z-akseli 3D-piirtämisessä) tai kiinnityspisteen. Osa ryhmistä on lukittu erityisiin toimintoihin, mutta esimerkiksi ryhmät 40-48 vaihtelevat riippuen objektista. Taulukossa 3 on kaikki kuvan 17 funktion tuomat arvot.

TAULUKKO 3. DXF-ryhmäkoodit.

DXF-group	Selitys
-1	APP: entity name
0	Text String entity type
5	Entity handle
10	Primary point
40-48	Double precision floating point values
50-58	Angles
67	Space – model/paperspace
100	Subclass datamarker
210	Extrusion direction
330-339	Soft-pointer handle
410-419	String

DXF-ryhmien määrä on huomattava ja arvot vaihtelevat -5 ja 1071 välillä (10). EntGet-funktio hakee ainoastaan tarpeelliset ja oletusarvoista poikkeavat entiteetit. Esimerkiksi kaikki DXF-ryhmät tuhannen jälkeen liittyvät extended data –muistiin (11).

Entiteettilistojen avulla tehty muokkaaminen on hyvin hidasta verrattuna komentoihin tai manuaaliseen muokkaamiseen. Ohjelmakoodin kirjoittamiseen kuluu todennäköisesti enemmän aikaa kuin pienien virheiden korjaamiseen. Miksi siis vaivautua tekemään monimutkaista ohjelmakoodia entiteettilistoilla? Entiteetteihin perustuvalla ohjelmakoodilla AutoLISP käy lävitse jokaisen objektin, jokaisen ominaisuuden, jokaisen arvon ja ylimääräiset data-alueet mukaan lukien. Yhdistämällä kaikki mahdollinen entiteettien kautta saatava tieto piirustuksesta voidaan valita äärimmäisen tarkasti halutut muutokset aivan mihin tahansa ominaisuuteen. Anekdoottina voidaan käyttää yrityksen haluavan tehdä muutoksen tuotteeseen tehtäviin reikiin. Reikäkoko muuttuu jokaisessa piirustuksessa M10:stä M12:een. Piirustuksien päivittäminen manuaalisesti tai komennoilla on hyvin työlästä verrattuna tietokoneen suorittamaan koodiin tai ohjelmaan, mikä tekee tarkoin valitut muutokset piirustuksiin sekunneissa. Ainoa tehokas tapa päivittää laajoja piirustustietokantoja on ohjelmallisesti tietokoneen avulla ja AutoLISP-ohjelmointikieli on yksi tapa. Tietokoneen avulla tehdyn muokkaamisen pelastuksena ja lankeamisena on kuitenkin mahdollisten virheiden määrä. Tietokone ei tee inhimillisiä virheitä piirustuksiin, mutta huonosti määritelty ohjelmakoodi voi muuttaa myös muita sattumalta samoilla arvoilla olevia objekteja. Esimerkiksi samoilla DXF-ryhmän arvoilla olevat objektit, mistä ainoastaan toista pitäisi muokata, tulevat kumpikin muokatuiksi.

5.3 Entiteettilistojen muokkaaminen ohjelmakoodilla

Tehdään käytännön esimerkki yleisestä entiteettilistoja käyttävästä ohjelmakoodista. Ohjelmakoodissa itsessään on kommentteja liittyen ohjelmakoodin kulkuun. Ohjelmakoodin toimintaperiaate on käydä kaikki entiteettilistat läpi ja tehdä halutut muutokset hyvin erityisiin osiin. Entiteettilistalta tarkastetaan jokaisen DXF-ryhmän arvot, jos nämä arvot täsmäävät niin näitä arvoja muokataan halu-

tuiksi. Esimerkki ohjelmakoodissa ympyröiden sädettä muutetaan arvosta 5 arvoon 10. Lisäksi muutettujen objektien entiteettidataan lisätään ylimääräistä tietoa dokumenttiin tehdyistä muutoksista.

Ohjelmakoodin alussa joudutaan alustamaan muutamia asioita. Funktion liittäminen kirjainyhdistelmään, applikaation rekisteröinti ja ensimmäisen entiteetin yhdistäminen muuttujaan ovat kaikki kertaluonteisia asioita. Itse työn tekevä osuus on kokonaan while-silmukan sisällä, minkä rikkoutumisen ehtona on entiteettien loppuminen ja palautettavan arvon muuttuminen epätodeksi, eli arvon ollessa *nil*. Silmukan sisällä ensimmäisenä asiana poimitaan E1 sisältämästä datasta halutut arvot ulos. Cdr-komento (12) liittyy listojen datan valintaan ja tämä kyseinen komento palauttaa kaikki listan arvot paitsi ensimmäisen. Assoc-komento (13) taas etsii listoista määritettyä argumenttia ja siihen liittyviä arvoja. Tässä tapauksessa muutettavan entiteetin DXF-ryhmän 40 arvo on (40 . 5.0), milloin RevData-muuttujaan tallennettava arvo on 5.0.

RevData-arvoa käytetään IF-funktion ehtona. Ympyrän säteen ollessa 5 ehto toteutuu ja AutoLISP käy läpi IF-funktion käskyt. Usein konditionaali-funktioiden jälkeen oletetaan käskyjen olevan ainoastaan yhden käskyn pituisia ja ohjelman jatkuvan välittömästi sen jälkeen. Esimerkiksi IF-funktion syntaksi tekee normaalisti toisen käskyistä, kuten esimerkistä nähdään (if *condition thenexpr [elseexpr]*). Tämä rajoitus voidaan ohittaa käyttämällä *progn*-komentoa (14) IF-funktion yhteydessä.

IF-funktion sisällä tehdään Extended Datan (Xdata) määrittelyt ja integrointi DXF-ryhmiin. Extdata-muuttujaan lisätään haluttu data listamuotoisena, kuten tässä ohjelmassa lisätään huomautus tehdystä muokkauksesta. Dataan voidaan lisätä vaikka muutokset ja päivämäärä ylimääräisenä piirustuksen sisäisenä dokumentaationa. Vanha entiteettilista ja uusi lisätty data yhdistetään takaisin samaan DXF-ryhmään append-komennolla (15) ja muutokset päivitetään piirustukseen entmod-komennolla (16).

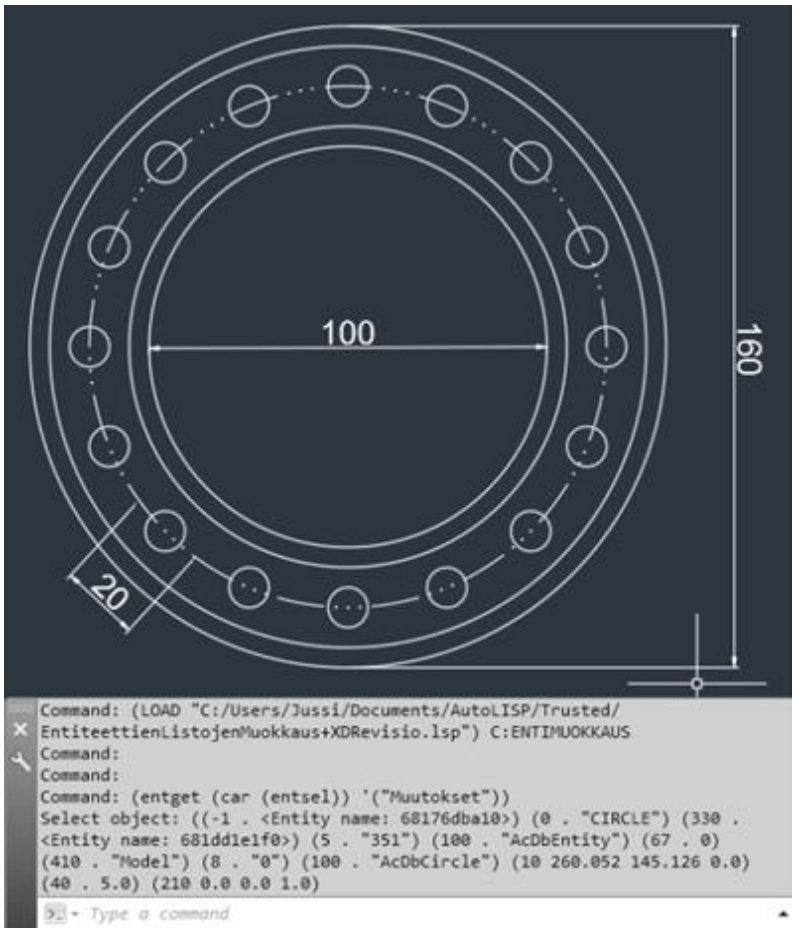
Itse ympyrän säteen muuttaminen on yksinkertaisempaa. Subst-komento (17) pitää sisällään sekä etsimis- että korvaus-komennot. Ensimmäinen argumentti

’(40 . 10.0) on korvaava arvo ja jälkimmäisempi argumentti ’(40 . 5.0) on korvattava arvo. Kummatkin edeltävistä argumenteista on annettu listamuodossa. Entiteettien päivittämisen jälkeen E1-muuttuja päivitetään piirustuksen seuraavalla entiteetillä. Komennot *entnext* (18) ja *entlast* (19) hakevat kuvasta seuraavan tai edellisen entiteetin. Silmukkaa suoritetaan kunnes kaikki piirustuksen entiteetit ovat käyty lävitse.

Alapuolella on koko ohjelmakoodi lyhyine kommentteineen. Ohjelmakoodin neljännellä rivillä on työkalu entiteettien tarkastamiseen. Syöttämällä käskyt komentoriville käyttäjä voi klikata objektin piirustusalueelta ja palautuksena tulee entiteettilista. Tällä voidaan tarkastaa onko objektin extended dataan lisätty muutoksista kertovaa dataa. Alapuolella on myös .lsp-tiedostoon lisätty entiteettien tarkastamiseen sopiva käsky.

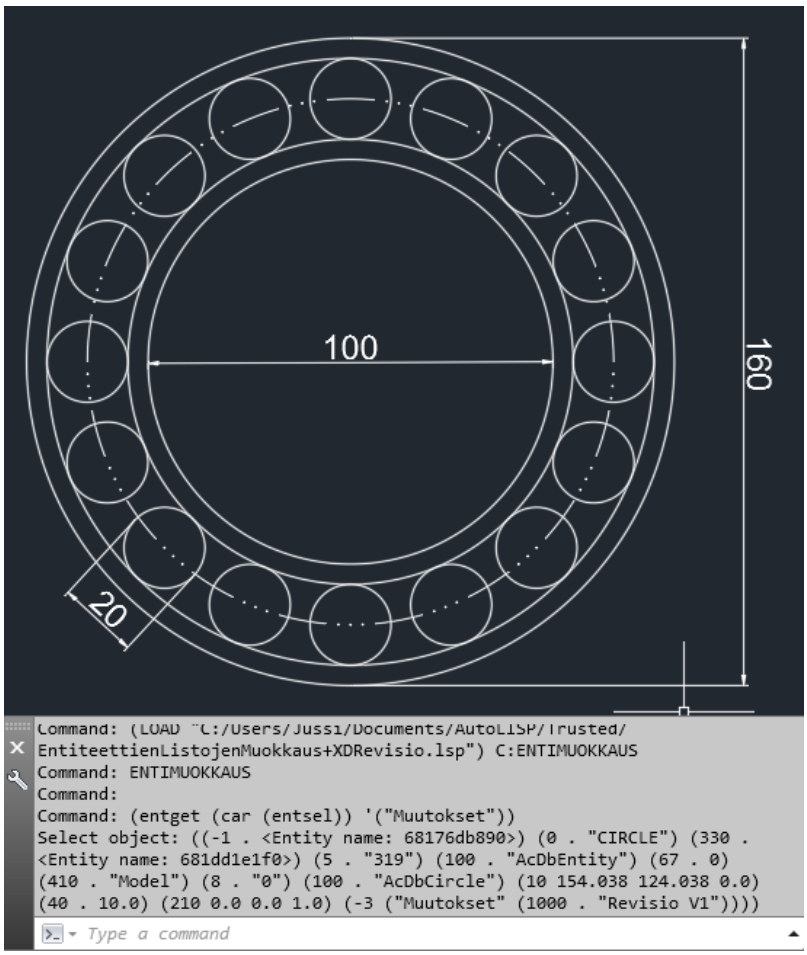
```
;;;Skripti käy kaikki entiteetit lävitse ja etsii halutulta DXF-ryhmältä haluttua arvoa.
;;;
;;;Työkalu Extended Datan tarkastamiseen
;;;                (entget (car (entsel)) '("Muutokset"))
;;;                ^^^^ Entiteettien tarkastamiseen ^^^^


(defun c:EntiMuokkaus ( / )
(regapp "Muutokset")                                    ;Applikaation rekisteröinti
(setq E1 (entnext))                                     ;Entiteetin alustaminen
(while E1                                               ;Silmukka kunnes entiteetit loppu
(setq RevData (cdr (assoc 40 (entget E1))))             ;Assoc hakee oikean DXF-ryhmän
(setq EnDa1 (entget E1))                                ;EnDa1 on muutettava entiteettidata
(if (= RevData 5.0)                                     ;Korvattavan arvon vienti IF-logiikalle
(progn                                                  ;Progn-komento laajalle IF-logiikalle
(setq lastent (entget E1))                                                    ;
(setq extdata '((-3 ("Muutokset" (1000 . "Revisio V1"))))) ;XDATA DXF-ryhmän muodost.
(setq entiUpd1 (append lastent extdata))                ;Entiteettilistojen yhdistäminen
(entmod entiUpd1)                                       ;Muutoksen päivitys
)
(princ)
)
(setq EnDa1 (subst '(40 . 10.0) '(40 . 5.0) EnDa1 ))    ;Arvojen korvaus EnDa1
(entmod EnDa1)                                          ;Muutoksen päivitys
(setq E1 (entnext E1))                                  ;Seuraavan entiteetin valinta
(princ)                                                 ;Hiljainen poistuminen
)
)
```

Testataan ohjelmakoodin toimivuutta esimerkkipiirustukseen. Kuulalaakerin kuulien sädettä muutetaan arvosta 5 arvoon 10. Piirustuksessa on läsnä myös muita erikokoisia ympyröitä häiriöiden testaamisen varalta. Kuvassa 18 on alkuperäinen piirustus entiteettilistoineen.



KUVA 18. Muokkaamaton kuva AutoCAD:ssa.

Ohjelmakoodin lataamisen jälkeen voidaan komentoriville kirjoittaa ”ENTIMUOKKAUS”, mikä kutsuu suoritettavaa AutoLISP-ohjelmakoodia. Ohjelmakoodin suorittamisen jälkeen voidaan taas valita yksi kuulalaakerin kuulista entiteettien tarkastukseen tehdyllä käskyllä. Kuvassa 19 on muokattu dokumentti entiteettilistoineen. Kuulien koko on muutettu ja ylimääräinen data on esillä entiteettilistassa.



KUVA 19. Muokattu kuva AutoCAD:ssa.

Kuten kuvasta 19 voidaan todeta, ovat DXF-ryhmän 40 data-arvot muuttuneet. Lisäksi uutena merkintänä on lisätty ylimääräinen data DXF-ryhmään 1000, mistä voidaan nähdä revisiot tai tehdyt muutokset.

6 MUUTOSTEN SUORITTAMINEN LAAJASTI

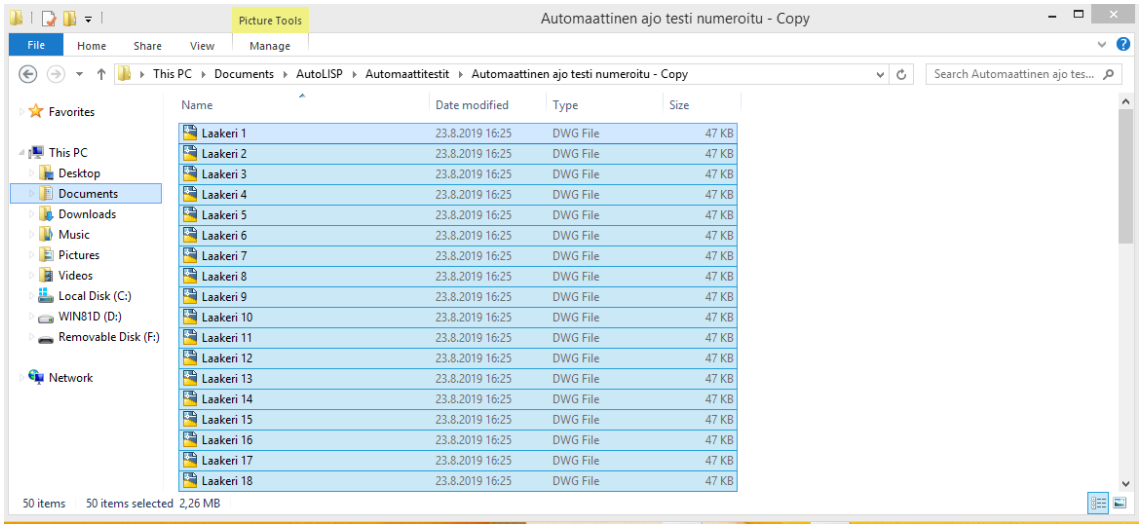
Aikaisemmin esiteltyjä ohjelmakoodeja on käytetty lähinnä yksittäisiin piirustuksiin. Tehtäessä muutoksia laajoihin tietokantoihin ohjelmakoodien lataaminen yksitellen piirustuksiin ja funktion kutsuminen komentorivillä vaikuttaa työläältä. Vaihtoehdoiksi piirustusten muokkaamiseen massoittain löytyy sekä AutoCAD:n sisältä että ulkopuolisista ohjelmista. Työtä on kuitenkin rajattu niin, ettei työssä käytetä ulkopuolisten tekijöiden ohjelmistoja tai ohjelmakoodeja liittyen automaattiseen lataamiseen tai siihen verrattaviin funktioihin.

Yksinkertainen tapa muuttaa piirustuksia massoittain on aikaisemmin mainittu tiedostojen nimeäminen. Haluttu ohjelmakoodi nimetään taulukon 1 olevien tiedostonimien mukaan ja muokataan sopivaksi. Käytetään esimerkkinä kohdan 5.3 EntiteettienListojenMuokkaus+XDRevisio –ohjelmakoodia. Ohjelmakoodin alkupuolelta ensinnäkin poistetaan funktion muodostaminen. Käyttäjältä ei vaadita mitään syötettä tai hyväksyntää toiminnoille, pois lukien mahdolliset virheet, koska tämä tulee hidastamaan piirustusten läpikäymistä huomattavasti. Lisäksi loppuun tulee lisätä piirustuksen tallentamiseen ja sulkemiseen liittyvät komennot. Lopuksi tiedoston nimi muutetaan *acaddoc.lsp* -nimiseksi. Muutettu ohjelmakoodi on luettavissa liitteessä 2.

Avattaessa mitä tahansa piirustusta AutoCAD lataa tiedoston *acaddoc.lsp* ja suorittaa sisällön välittömästi. Tämä poistaa AutoCAD:n normaalin suunnittelun käytettävyyden käytännössä kokonaan, paitsi muutettavien piirustusten osalta. Ohjelma käy läpi nopeasti entiteettilistat, tekee muutokset, tallentaa & sulkee piirustuksen. Sama prosessi käydään läpi vaikka piirustus olisi juuri luotu tai ei liittyisi millään tavalla muokkaukseen. Muokattavien piirustusten kohdalla voidaan yksinkertaisesti avata tiedostoja ja AutoLISP sulkee niitä automaattisesti muokkausten jälkeen.

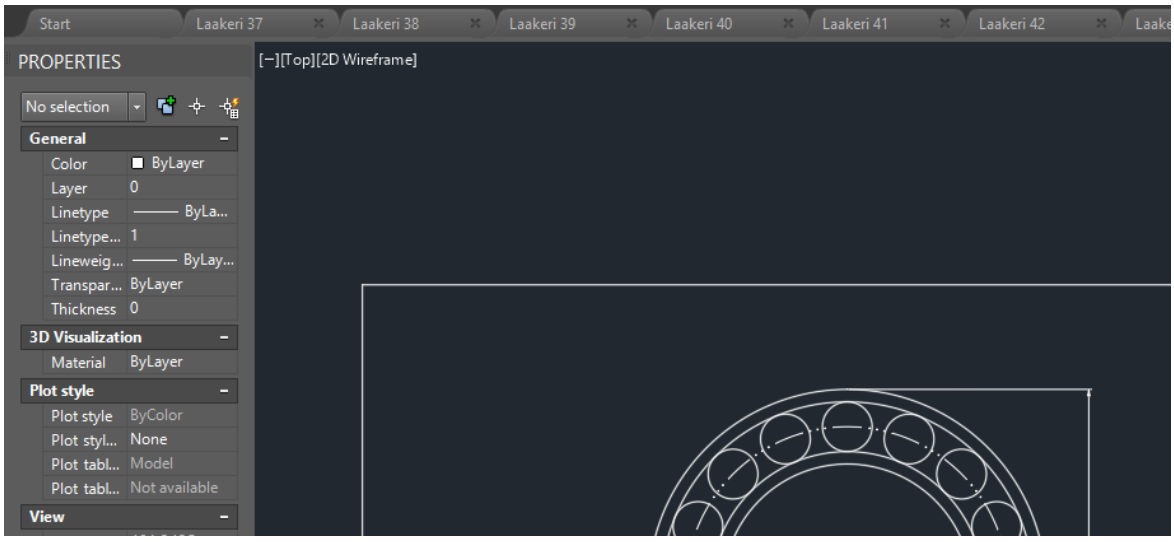
Ohjelmakoodin tehdessä kaiken automaattisesti käyttäjälle jää ainoastaan uusien piirustusten avaaminen AutoCAD:in. Muutaman piirustuksen avaaminen ei sinänsä vaadi lisätoimenpiteitä, mutta kun muutettavien piirustusten määrä lähestyy satoja, kannattaa miettiä jo vaihtoehtoja. Yksinkertaisimpana vaihtoehtona on

valita muutettavat tiedostot resurssienhallinnassa ja avata tai raahata ne AutoCAD:in. Piirustukset käydään järjestyksessä ylhäältä alas, joten nimeämiskäytännöt kannattaa ottaa huomioon tehtäessä esimerkiksi sivunumerointia. Esimerkiksi kuvassa 20 on valittu 50 muutettavaa piirustusta.



KUVA 20. Muutettavat piirustukset resurssien hallinnassa.

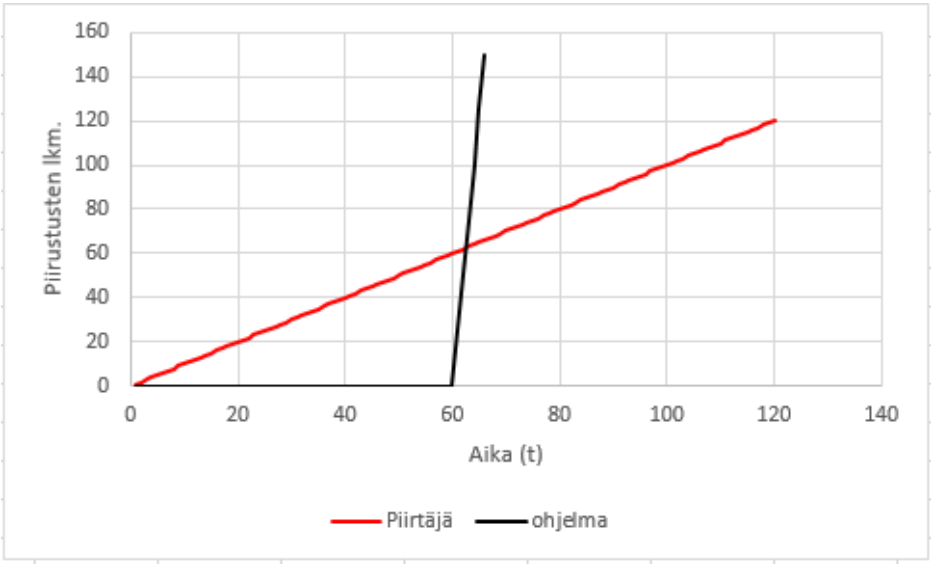
Raahaamalla nämä piirustukset AutoCAD:n piirustustilaan, saadaan avattua piirustuksia laajalla skaalalla. Yhdistettäessä tämä aikaisemmin mainittuihin automaattisesti piirustusta avattaessa ladattaviin AutoLISP-tiedostoihin, kuten acaddoc.lsp, voidaan ajaa muutoksia massoittain suurillekin tietokannoille. Kuvassa 21 on avattu piirustukset ilman ohjelmakoodia, koska muutoin välilehdissä olevat piirustukset suljettaisiin muutaman sekunnin sisällä avaamisesta.



KUVA 21. Piirustusten avaaminen laajalla skaalalla.

Ohjelmallisen muutosten tekemisen edut tulevat esille määrien kasvaessa suuriksi. Esimerkkinä käytetystä laakerista muutetaan 16 ympyrää jokaisessa dokumentissa, mikä tarkoittaa manuaalisesti muokattaessa 16 objektin valitsemista tai klikkaamista. Tämän jälkeen käyttäjän tulisi käydä ominaisuuksista vaihtamassa ympyrän säde kaikille objekteille. Lopuksi käyttäjä tallentaa ja sulkee kuvan. Edellä mainittuihin toimenpiteisiin vaaditaan käyttäjältä ainakin parikymmentä klikkausta yhteensä ja näppäimistöltä useampaa syötettä, kuten arvojen muuttaminen ja näppäinyhdistelmät erilaisille komennoille. Vaadittujen klikkausten määrä kertaantuu nopeasti dokumenttien määrän kasvaessa ja esimerkissä käytetty 50 dokumentin muokkaaminen vaatii käyttäjältä vähintään tuhatta klikkausta AutoCAD:in sisällä, puhumattakaan ohjelmiston ulkopuolella tehtävistä toimenpiteistä. Ohjelmakoodilla ajettuna 50 piirustuksen prosessointiin menee noin kahdesta kolmeen minuuttia, riippuen tietokoneen laskentatehoista.

Manuaalisen ja ohjelmallisen muokkaamisen ajallisia eroja käsiteltäessä dokumenttien määrän tärkeys korostuu entisestään. Edellä mainittuihin manuaalisiin toimenpiteisiin kuluu suunnittelijalta esimerkiksi keskimäärin minuutti per dokumentti. Tämä aikamääre on puhtaasti riippuvainen tehtävien muutosten laadusta, mutta tässä tapauksessa käytetään yhden minuutin keskimääräistä arvoa. Ohjelmallisesti muutosten tekemisessä kului kaikilta 50 dokumentilta kaksi minuuttia. Kuviossa 1 on visualisoitu manuaalisen ja ohjelmallisen muokkaamisen ero. Ohjelman tekemiseen on oletettu kuluvan tunti, joten tänä aikana suunnittelijan työpanos on sidottuna ohjelmointiin.



KUVIO 1. Manuaalisen ja ohjelmallisen muokkaamisen eroja.

7 POHDINTA

Viimeisin suuri muutos suunnittelun alalla oli tietokone ja tietokoneavusteisen suunnittelun käyttöönotto. Tämän jälkeen ohjelmistoihin on tullut lukemattomia päivityksiä ja käytettävyys ensimmäisistä ohjelmistoversioista on parantunut huomattavasti. AutoLISP lisättiin AutoCAD:in rinnalle jo hyvin varhaisessa 2.1 versiossa ja on siitä lähtien ollut käytettävissä, mutta silti osaajien määrän voi sanoa Suomen kokoisessa maassa olevan hyvin vähäistä ja osaaminen on itsenäisesti hankittua. Ohjelmallinen teknisten dokumenttien ja piirustusten muokkaamisen toteuttamiseen vaaditaan useimmiten suuria määriä sähköisiä dokumentteja, kuten suuret laitosprojektit, suunnittelutoimistot tai valmistava teollisuus. Lisäksi tehtyjen muutosten tulee olla, kuten aikaisemmin on mainittu, hyvin kaavamaisia ja toistuvia. Yksityiskohtaisempaa ja räätälöityä suunnittelua tehtäessä AutoLISP-kielellä voidaan tehdä piirtämistä helpottavia funktioita. Käyttäjän luovuus määrittää miten tehokkaasti AutoLISP-kieltä voidaan käyttää yksityiskohtaisessa ja massoittain tehtävässä piirtämisessä.

Teknisen suunnittelun automatisoinnissa ja automatisoinnissa yleensäkin pitäisi kiinnittää huomiota erilaisiin muuttujiin. Onko tehtävänä oleva työ kertaluontoista vai onko nähtävissä toistuvia tai kaavamaisia toimia? Tehtävien ollessa vähäisiä piirustusten ja muutosten osalta, manuaalisesti tehty muokkaaminen on nopeampaa ja suoraviivaisempaa kuin ohjelmallinen muokkaaminen. Olennaiseksi asiaksi ohjelmallisen muokkaamisen saralla muodostuu muokattavien objektien tai piirustusten määrä. Kuviosta 1 kävi ilmi ohjelmallisen muokkaamisen huomattava nopeusero tehtäessä mekaanisia toimintoja. Valmiita ohjelmakoodeja käytettäessä ohjelmointiin ei kulu ollenkaan aikaa ja prosessi nopeutunee entisestään. Ohjelmalliseen muokkaamiseen liittyy vielä yksi iso ero manuaaliseen muokkaamiseen verrattuna. Ohjelman voi jättää taustalle pyörimään ja tehdä jotain muuta samalla kun ohjelma suorittaa rutiinejansa. Tämä vapauttaa suunnittelijan vähäpätöisemmistä töistä muihin enemmän luovuutta vaativiin toimiin.

LÄHTEET

1. Ambrosius Lee. 2015. AutoCAD Platform Customization: AutoLISP. John Wiley & Sons.

2. Autodesk, inc. 2012. AutoLISP Developer's Guide.

3. Finkelstein Ellen. 2014. AutoCAD 2015 and AutoCAD LT 2015 Bible. John Wiley & Sons.

4. ESET LCC. N.d. ACAD/Medre.A. [online][viitattu 24.7.2019] www.eset.com/sg/about/newsroom/press-releases1/whitepapers/acadmedrea-1000s-of-autocad-designs-leaked-in-suspected-industrial-espionage/

5. Harkow Roy. 1995. Essential AutoLISP®. Springer Publishing Company.

6. Autodesk, inc. 2015. OSMODE (System Variable). knowledge.autodesk.com/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/AutoCAD-Core/files/GUID-DD9B3216-A533-4D47-95D8-7585F738FD75-htm.html

7. Autodesk, inc. 2014. AutoLISP and VBA Security Controls in AutoCAD 2013 SP1. knowledge.autodesk.com/support/autocad/troubleshooting/caas/sfdcarticles/sfdcarticles/AutoLISP-and-VBA-Security-Controls-in-AutoCAD-2013-SP1.html

8. Autodesk, inc. 2015. SECURELOAD (System Variable). knowledge.autodesk.com/support/autocad/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/AutoCAD-Core/files/GUID-541566C6-2738-49DD-87C3-C1490E924A02-htm.html

9. Autodesk, inc. 2015. entget (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-12540DAE-C84B-4BDB-AEEC-DDFE5BE3C42A-htm.html

10. Autodesk, inc. 2017. DXF Group Codes in Numerical Order Reference. knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2018/ENU/AutoCAD-DXF/files/GUID-3F0380A5-1C15-464D-BC66-2C5F094BCFB9-htm.html

11. Autodesk, inc. 2015. About Attaching Extended Data to an Entity (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2016/ENU/AutoCAD-AutoLISP/files/GUID-FA8BBFB8-5C3E-4742-A3A2-CBCDB168FB08-htm.html

12. Autodesk, inc. 2015. cdr (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-F9CD8FF3-022A-4323-BAE7-390174451537-htm.html

13. Autodesk, inc. 2015. assoc (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-46309786-DAF6-4C28-8448-599FBC8A4F6A-htm.html

14. Autodesk, inc. 2015. progn (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-6B2DD269-858D-4C01-ABDB-765DD08284FE-htm.html

15. Autodesk, inc. 2015. append (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-952B175A-D565-43A4-9208-0E6A27A5E742-htm.html

16. Autodesk, inc. 2015. entmod (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-C7D27797-247E-49B9-937C-0D8C58F4C832-htm.html

17. Autodesk, inc. 2015. subst (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-25214E69-090A-45C3-8210-6D9801255E44-htm.html

18. Autodesk, inc. 2015. entnext (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-65924CF5-0C51-4E36-8B38-7A5513951A04-htm.html

19. Autodesk, inc. 2015. entlast (AutoLISP). knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/AutoCAD-AutoLISP/files/GUID-75DBA9B2-034B-4377-A4E2-21D37B298D86-htm.html

LIITTEET

Liite 1. Entiteettilistojen muokkaamisen ohjelmakoodi

(1)

```
;;;;Entiteettilistojen muokkaus
;;;Skripti käy kaikki entiteetit lävitse ja etsii halutulta DXF-ryhmältä haluttua ar-
voa.
;;;
;;;Työkalu Extended Datan tarkastamiseen
;;;             (entget (car (entsel)) '("Muutokset"))
;;;             ^^^^ Entiteettien tarkastamiseen ^^^^


(defun c:EntiMuokkaus ( / )
(regapp "Muutokset")
(setq E1 (entnext))

(while E1
(setq RevData (cdr (assoc 40 (entget E1))))
             (setq EnDa1 (entget E1))
(if (= RevData 5.0)
(progn
             (setq lastent (entget E1))
                          ;
             (setq extdata '((-3 ("Muutokset" (1000 . "Revisio V1")))))
             (setq entiUpd1 (append lastent extdata))
             (entmod entiUpd1)
(princ)
)
             (setq EnDa1 (subst '(40 . 10.0) '(40 . 5.0) EnDa1 ))
             (entmod EnDa1)
             (setq E1 (entnext E1))
             (princ)

)
)
```

Liite 2. Entiteettilistojen muokkaamisen ohjelmakoodi automatisoituna.

```
;;;;Entiteettilistojen muokkaus
;;;Skripti käy kaikki entiteetit lävitse ja etsii halutulta DXF-ryhmältä haluttua ar-
voa.
;;;
;;;Työkalu Extended Datan tarkastamiseen
;;;             (entget (car (entsel)) '("Muutokset"))
;;;             ^^^^ Entiteettien tarkastamiseen ^^^^


 (regapp "Muutokset")
(setq E1 (entnext))

(while E1
(setq RevData (cdr (assoc 40 (entget E1))))
             (setq EnDa1 (entget E1))
(if (= RevData 5.0)
(progn
             (setq lastent (entget E1))
                          ;
             (setq extdata '((-3 ("Muutokset" (1000 . "Revisio V1")))))
             (setq entiUpd1 (append lastent extdata))
             (entmod entiUpd1)
(princ)
)
             (setq EnDa1 (subst '(40 . 10.0) '(40 . 5.0) EnDa1 ))
             (entmod EnDa1)
             (setq E1 (entnext E1))
             (princ)

)
(command "_.save" "")
(command "_.close" "")
```