გ ე გ მ ა

1. მიმოხილვა
   ა) ავტომატიზებული საკონსტრუქტორო ტექნოლოგიური დაპროექტება.
   ბ) ადს
   ც) ადს–ის არქიტექტურები (ხაზგასმა ღია არქიტექტურებზე)

2. AutoCAD–ის მიმოხილვა (შიდა მოდულების თვალსაზრისით)
   ACIS Kernel, Windows, დაპროგრამების გარემოებები

3. AutoCAD–ის დაპროგრამების მეთოდების მიმოხილვა.
   ა) Lisp მიმოხილვა.
   ბ) Basic მიმოხილვა.
   ც) ObjectARX მიმოხილვა.

4. საკვლევი ამოცანის ფორმულირება.

5. "პრეპროცესორის" სისტემის აღწერა.

შესავალი

პროექტირება, როგორც პროცესი, განიხილება როგორც გადასვლის პროცესი, ტექნიკური ამოცანით განსაზღვრული, ახალი ფუნქციების შესრულების მოთხოვნათა ფორმირებიდან იმ ობიექტის აღწერის შექმნამდე, რომელმაც უნდა შეასრულოს ეს ფუნქციები. მიღებული აღწერა საკმარისი უნდა იყოს საწარმოო პროცესის დაგეგმისათვის და წარმოების ორგანიზებისათვის. პროექტირების მიზნის მიღწევა ხორციელდება არსებული რესურსების გათვალისწინებით. ანუ ხდება ისეთი ფაქტორების გათვალისწინება როგორიცაა: წარმოების მაჭტაბი, ენერგომოხმარება, საჭირო ტექნიკური დანადგარებისა და აღწურვილობის არსებობა, შეზღუდვა წარმოების პროცესის და თავად შესაქმნელი ობიექტის ფასზე და თვით პროექტირების ვადა.

პროექტირება, როგორც პროცესი, იყოფა რამოდენიმე ეტაპად. ძირითადად შეიძლება ჩაითვალოს ისეთი ეტაპები, როგორიცაა ტექნიკური ამოცანის განსაზღვრა და ანალიზი, შესაქმნელი ობიექტის ესკიზის შექმნა, ობიექტის გეომეტრიული და ფიზიკური პარამეტრების განსაზღვრა, საპროექტო დოკუმენტაციის შედგენა. შემდგომ და საკმაოდ მნიშვნელოვან ეტაპად ითვლება ტექნოლოგიური პროცესის დაპროექტემა. ზემოთ თქმულიდან გამომდინარე პროექტირების პირველ ეტაპზე პროექტირების ამოცანას წარმოადგენს შესაქმნელი ობიექტის აღწერის შექმნა/დაპროექტება, ანუ დაპროექტების ობიექტს წარმოადგენს შესაქმნელი ობიექტი, ხოლო მეორე ეტაპზე უკვე ხდება საწარმოო პროცესის დაპროექტება, ანუ დაპროექტების ობიექტი უკვე საწარმოო პროცესია.

თანამედროვე ტექნიკურ მოთხოვნათა ზრდასთან ერთად სულ უფრო რთული ხდება პროექტირების პროცესის ორგანიზება. მაგალითად რთული დეტალების დაპროექტებისათვის და მათი დამზადებისთვის საჭირო ტექნიკური პროცესის დაპროექტებისათვის საჭიროა დამპროექტებელთა ფართო ჯგუფის მიერ საკმაოდ

დიდი და შრომატევადი სამუშაოს ჩატარება, დიდი რაოდენობით ტექნიკური ნახაზებისა და ტექნიკური დოკუმენტაციის შექმნა. აქედან გამომდინარე, შრომის ოპტიმიზაციის მიზნით, სულ უფრო ხშირად გამოიყენება ავტომატიზებული დაპროექტება.

ავტომატიზებული დაპროექტებისათვის გამოიყენება კომპიუტერი. კომპიუტერის გამოყენება საგრძნობლად ზრდის მწარმოებლის მუშაუნარიანობას, ხელს უწყობს პროექტირების ხარისხის გაუმჯობესებას, აუმჯობესებს დამპროექტებელთა შორის ინფორმაციულ კავშირს და რაც მთავარია ხელს უწყობს საკონსტრუქტორო ტექნოლოგიური დაპროექტების მონაცემთა და ცოდნის ბაზის შექმნას. საკონსტრუქტორო ტექნოლოგიური დაპროექტებია ხორციელდება რამდენიმე ეტაპად, ამასთან პროექტირების ობიექტი თავდაპირველად წარმოადგენს თვითონ ნაკეთობას, მის კონსტრუქციას, ხოლო შემდგომ ეტაპებზე პროექტირების ობიექტი უკვე წარმოადგენს არა თვითონ ნაკეთობას, არამედ ამ ნაკეთობის დასამზადებლად საჭირო წარმოების პროცესს. როგორც ზემოთ ავღნიშნეთ საკონსტრუქტორო ტექნოლოგიური დაპროექტება, როგორც პროცესი ხორციელდება რამდენიმე ეტაპად:

I. **წინა საპროექტო კვლევა.**

ამ ეტაპზე ხდება ბაზრის შესწავლა ობიექტის დამზადებაზე მოთხოვნილების შესწავლის მიზნით და/ან ობიექტის გაუმჯობესების მოთხოვნილების შესწავლის მიზნით.

II. **გეომეტრიული მოდელირება.**

ამ ეტაპზე ხდება დასამზადებელი ობიექტის გეომეტრიული მოდელის შექმნა კომპიუტერის საშუალებით, იმ მიზნით ,რომ მოხდეს ობიექტის გეომეტრიული პარამეტრების დეტალური განსაზღვრა.

III. **ფუნქციონალური მოდელირება.**

ამ ეტაპზე ხდება დასამზადებელი ობიექტის ვირტუალური გეომეტრიული მოდელისათვის მასალის, ზედაპირების სიგლუვისა და სიმქისის დადგენა.

IV. **საინჟინრო ანალიზი.**

საინჟინრო ანალიზი იტერაციული პროცესია, რომელიც შეიძლება დავყოთ ორ ძირითად ეტაპად:

<u>ვირტუალური პროტოტიპირება</u> და <u>ფიზიკური პროტოტიპირება</u>.

ვირტუალურ პროტოტიპირება თავისთავად შეიცავს შემდეგ ქვეეტაპებს:

1. <u>ვიზუალიზაცია</u>.

   ვიზუალიზაცია გამოიყენება ვირტუალურად შექმნილი ობიექტის ერგონომიული და ვიზუალური პარამეტრების დათვალიერებისათვის და დიზაინერული იდეებისთვის რეალიზირებული, მოცულობითი გამოსახულების მისაღებად. ვიზუალიზაცია დამყარებულია სხეულების მოდელირებისა და რენდერინგის ტექნოლოგიაზე, რომელიც ვირტუალური მრავალი ხედვის წერტილის, განათების სხვადასხვა საშუალებების, გამჭირვალობის და ტექსტურების გამოყენების საშუალებას იძლევა

2. <u>მასური ანალიზი</u>.

   მასური ანალიზი ვირტუალური პროტოტიპირების ერთერთი ეტაპია, რომლის დროსაც ხდება ვირტუალურად შექმნილი გეომეტრიული მოდელისათვის ისეთი ფიზიკური თვისებების შერჩევა / მოდიფიცირება, როგორიცაა მაგალითად მასა, წონა, სიმზიმის ცენტრი და სხვა.

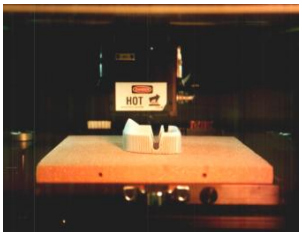
3. <u>დინამიკაში ანალიზი</u>. დინამიკაში ანალიზი მოძრაობის სიმულაციის და ვიზუალიზაციის საშუალებას იძლევა რომლის დროსაც ხდება დასამზადებელი ობიექტის დინამიური მახასიათებლების ანალიზი. (ვირტუალურად, კომპიუტერის საშუალებით).
4. <u>ანალიზი სასრული ელემენტების მეთოდით</u>.

   ამ ეტაპზე დასამზადებელი ობიექტის ვირტუალური მოდელი იფარება ეგრეთ წოდებული კუბური ბადით. ამ ბადის შემადგენელ თითოეულ კუბს "სოლვერი" ეწოდება და ის, მათემატიკური თეორიის მიხედვით, წარმოადგენს შესავალი და გამოსავალი პარამეტრების ერთმანეთთან დამოკიდებულების ფუნქციას. სასრული ელემენტების მეთოდის გამოყენებით შესაძლებელი ხდება ისეთი თვისებების ანალიზი, როგორიცაა სითბოგამტარობა, დრეკადობა და სხვა ფიზიკური თვისებები.

ფიზიკური პროტოტიპირება კი თავისთავად შეიცავს შემდეგ ქვეეტაპებს:

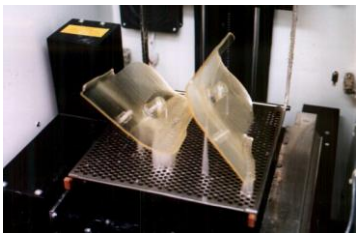
1. <u>სტერეოლიტოგრაფიის მეთოდი</u>.

   სტერეოლიტოგრაფიის ძირითად სამუშაო ელემენტს წარმოადგენს ლაზერის სხივი, რომელიც ახდენს, დასამზადებელი ობიექტის კვეთის თანმიმდევრულ "შემოხაზვას" სინათლისმიმართ მგრძნობიარე პოლიმერული სითხის ზედაპირზე. ეს სითხე მაგრდება მხოლოდ იმ ადგილას სადაც მას ეხება ლაზერის სხივი. შემდგომ ხდება შემდეგი კონტური შემოხაზვა ლაზერის სხივის მიერ და ეს პროცესი გრძელდება ავტომატურად დეტალის სრული კონტურის მიღებამდე.



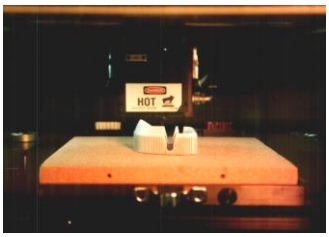
2. <u>ლამინირებით მიღებული პროტოტიპების შექმნა</u>.

ამ მეთოდის დროს ხდება დასამზადებელი ობიექტის გარკვეული სისქის კვეთებად დაყოფა. ამ კვეთების რიგრიგობით დამზადება ქაღალდისგან ან პლასტმასისგან და მათი ლამინირების მეთოდით ერთმანეთთან თანმიმდევრული შეერთება სამგანზომილებიანი მოდელის მისაღებად.


3. <u>ლაზერის სხივით გამოწვის მეთოდი.</u>

ამ მეთოდის დროს ლაზერის სხივით ხდება სპეციალური ფხვიერი ნივთიერებიდან დასამზადებელი სხეულის კვეთების თანმიმდევრული გამოწვა. თავდაპირველად ხდება პირველი კვეთის გამოწვა, შემდეგ მეორე კვეთის და ასე შემდეგ მათი ერთმანეთთან ავტომატურად მიერთებით სამგანზომილებიანი პროტოტიპის მიღებამდე.



როგორც ზემოთ ავღნიშნეთ, საინჟინრო ანალიზი ეს არის ეტაპი რომლის დროსაც ხორციელდება საპროექტო გადაწყვეტის მიღება და ამ ეტაპის ამოცანები დაიყვანება სინთეზისა და ანალიზის ამოცანებამდე. საპროექტო გადაწყვეტილების

მიღების პროცედურები იტერაციული ხასიათისაა. ეს იმას ნიშნავს, რომ თავდაპირველად ხორციელდება გადაწყვეტილების, შემდეგ კი მიღებული შედეგების ანალიზი. ამის შემდეგ ხორციელდება ისევ სინთეზი ანალიზის შედეგად მიღებული შედეგების გათვალისწინებით, შემდეგ ისევ ანალიზი და ა.შ.

V. **ნახაზის შედგენა / დოკუმენტაციის გაფორმება.**

ამ ეტაპზე ხდება დასამზადებელი ობიექტის ნახაზის შედგენა, საპროექტო გადაწყვეტილები გაფორმება, დოკუმენტაციის შედგენა. გეომეტრიული მოდელისგან განსხვავებით ნახაზი დაიტანება ფურცელზე, გამოყვანა/დოკუმენტირების ტექნიკური უზრუნველყოფის საშუალებებით (პრინტერი, პლოტერი), ანუ მას უკვე ტექნიკური დოკუმენტის სახე აქვს.

VI. **ტექნოლოგიური პროცესის დაგეგმვა.**

ამ ეტაპის ამოცანას წარმოადგენს ტექნოლოგიური პროცესის გეგმის შედგენა. ამ დროს ისაზღვრება ტექნოლოგიური ოპერაციები, მათი შესრულების თანმიმდევრობა, ხდება ნამზადის შერჩევა, დასამუშავებელი ინსტრუმენტების შერჩევა, ჭრის რეჟიმების შერჩევა და ა.შ.

VII. **მმართველი პროგრამის გენერაცია.**

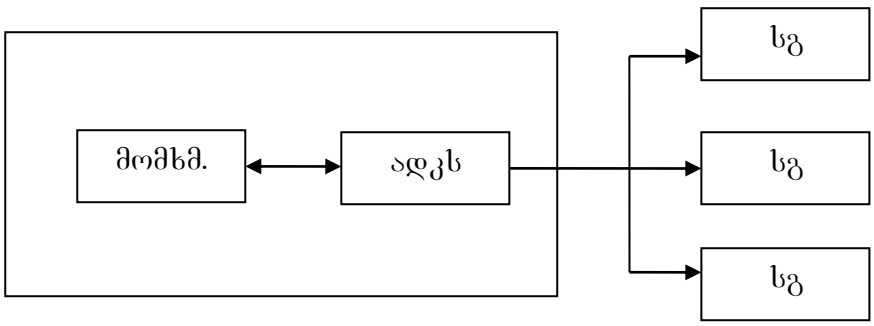
ამ ეტაპზე ხდება ინსტრუმენტის გადაადგილების ტრაექტორიის გეომეტრიული პარამეტრების განსაზღვრა, ტექნოლოგიური დანადგარის მუშაობის რეჟიმების განსაზღვრა და ამ დანადგარზე ტექნოლოგიური პროცესების მიმდინარეობის პროგრამირება და როგორც შედეგი მმართველი პროგრამის ფორმირება.

VIII. **მმართველი პროგრამის გამართვა.**

ამ ეტაპზე ხორციელდება მმართველი პროგრამის ტესტირება / გამართვა. ეს შესაძლებელია განხორიელდეს ორი მეთოდით: კომპიუტერული მოდელირების საშუალებით და ეგრეთწოდებული საცდელი გავლების მეთოდით. კომპიუტერული მოდელირების დროს ხდება ჭრის პროცესის ვირტუალური იმიტაცია კომპიუტერისა და შესაბამისი პროგრამის გამოყენებით, ხოლო საცდელი გავლების მეთოდით მართველი პროგრამის გამართვის პროცესში ხდება საწარმოო ჩარხზე ჭრის პროცესის იმიტაცია ნამზადის გარეშე.

ავტომატიზებული დაპროექტების სისტემები (ადს)

ავტომატიზებული დაპროექტების სისტემა (ადს) განისაზღვრება როგორც ორგანიზაციულ-ტექნიკური სისტემა, რომელიც შედგება დამპროექტებელი ორგანიზაციის ქვეგანყოფილებისგან (მომხმარებლისგან) და ავტომატიზებული დაპროექტების კომპლექსური საშუალებებისგან (კომპიუტერისაგან).



ნახ.№1 ადს–ის არქიტექტურა

ავტომატიზებული დაპროექტების სისტემებში (ადს) დამპროექტებელი და "კომპიუტერი" ურთიერთქმედებენ ე.წ ინტერაქტიულ რეჟიმში, ანუ დიალოგურ რეჟიმში. როდესაც დამპროექტებელი აძლევს დირექტივას "კომპიუტერს", ხოლო კომპიუტერი ახორციელებს მის შესრულებას.

ინტერაქტიულ რეჟიმში დამპროექტებელსა და "კომპიუტერს" შორის ფუნქციები განაწილებულია შემდეგნაირად: "კომპიუტერი" ასრულებს გამოთვლით ოპერაციებს, გამოყავს შესაძლო ალტერნატიული ვარიანტები, დამპროექტებელი კო ახორციელებს გადაწყვეტილების მოღებას.

როგორც ზემოთ ავღნიშნეთ ტერმინ "კომპიუტერ"–ში არ იგულისხმება მარტო პერსონალური კომპიუტერი, ეს არის ავტომატიზებული დაპროექტების კომპლექსური სისტემების ერთობლიობა. ავტომატიზებული დაპროექტების კომპლექსური სისტემები შეიცავენ შემდეგ საშუალებებს (უზრუნველყოფებს):

I. მათემატიკური უზრუნველყოფა.

II. პროგრამული უზრუნველყოფა.

III. ინფორმაციული უზრუნველყოფა.

IV. ტექნიკური უზრუნველყოფა.

V. ლინგვისტიკური უზრუნველყოფა.

განვიხილოთ თითოეული უფრო ვრცლად:

მათემატიკური უზრუნველყოფა. ეს არის ალგორითმები, რომლის მოხედვითაც მუშავდება ავტომატიზებული დაპროექტების პროგრამული უზრუნველყოფა. მათემატიკური უზრუნველყოფის სხვა და სხვა ელემენტებს შორის არსებობს ფუნქციონალური მოდელის აწყობის ინვარიანტული ელემენტი-პრინციპები, ალგებრული და დიფერენციალური ტოლობების გამოთვლის მეთოდები,

ექსტრემალური ამოცანების გადაწყვეტის საშუალებები. მათემატიკური უზრუნველყოფის შექმნის ეტაპი არის ერთერთი ურთულესი ეტაპი ავტომატიზებული დაპროექტების სისტემის შექმნის დროს. ადს–ის მწარმოებლურობა და მუშაობის ეფექტურობაც, ყველაზე მეტად, სწორედ ამ ეტაპზეა დამოკიდებული.

პროგრამული უზრუნველყოფა. ადს–ის პროგრამული უზრუნველყოფა წარმოადგენს ყველა იმ პროგრამის და ამ პროგრამების საექსპლუატაციო დოკუმენტაციის ერთობლიობას, რომელიც საჭიროა ავტომატიზებული დაპროექტებისათვის. პროგრამული უზრუნველყოფა იყოფა საერთო სისტემურ და სპეციალურ პროგრამულ უზრუნველყოფად. საერთო პროგრამული უზრუნველყოფა განკუთვნილია ტექნიკური საშუალებების ფუნქციონირების ორგანიზირებისათვის ანუ გამოთვლითი პროცესის დაგეგმვისა და მართვისათვის, არსებული რესურსების განაწილებისათვის. სპეციალური პროგრამული უზრუნველყოფაში კი ხორციელდება მათემატიკური უზრუნველყოფები კონკრეტულად პროექტის პროცედურების შესასრულებლად.

ტექნიკური უზრუნველყოფა წარმოადგენს ტექნოლოგიურ საშუალებების ურთიერთდაკავშირებულ და ურთიერთმოქმედ ერთობლიობას, განკუთვნილი ავტომატიზებული პროექტირების შესასრულებლად, ტექნიკური უზრუნველყოფა იძლევა შემდეგის საშუალებას: მონაცემების პროგრამული დამუშავება, მონაცემების მომზადება და შეტანა, ასახვა და დოკუმენტირება, საპროექტო გადაწყვეტის არქივი, მონაცემების გადაცემა. ტექნიკური უზრუნველყოფა შეიძლება დავყოთ ოთხ ქვეჯგუფად:

1. **შეყვანის.** (სკანერი, დიგიტაიზერი და სხვა)

   შეყვანის ტექნიკური უზრუნველყოფების საშუალებებით ხდება ქაღალდზე არსებული ნახაზის შეყვანა კომპიუტერში, ანუ ციფრულ ფორმატში გადაყვანა მისი რედაქტირების ან შენახვის მიზნით.

2. **გამოყვანა/დოკუმენტირების.** (პრინტერი, პლოტერი და სხვა)

   გამოყვანა/დოკუმენტირების ტექნიკური უზრუნველყოფების საშუალებებით ხდება კომპიუტერში არსებული (ვირტუალურად შექმნილი) გეომეტრიული მოდელის ნახაზის და დოკუმენტაციის ამობეჭდვა ფურცელზე.

3. **გადამუშავების.** (პროცესორი)

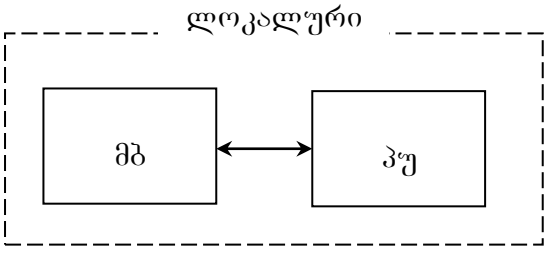
   კომპიუტერში ვირტუალურად ობიექტების შექმნისას, მათი გეომეტრიული ზომების რედაქტირებისას, ინფორმაციის გადამუშავება ხდება პროცესორის მიერ.

4. **შენახვის** (ვინჩესტერი, კისკეტები, კომპაკტდისკები და სხვა)

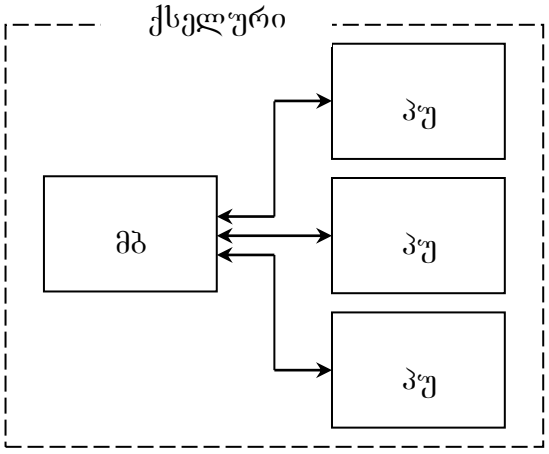
   კომპიუტერის მეშვეობით შექმნილი ობიექტის პროექტი ინახება კომპიუტერის ვინჩესტერში, დისკეტებზე, კომპაქტდისკებზე და სხვა ინფორმაციის შემნახველ მოწყობილობებში იმ მიზნით, რომ დამპროექტებელს შეეძლოს ამ პროექტის რედაქტირება, კოპიის შექმნა, ამობეჭდვა ქარალდზე, რაც საგრძნობლად ამცირებს დამპროექტებლის შრომას.

ლინგვისტიკური უზრუნველყოფის საფუძველს წარმოადგენს სპეციალური პროგრამირების ენები, რომელთა დანიშნულებაა ავტომატიზებული დაპროექტების პროცედურების პროგრამული აღწერა. ლინგვისტიკური უზრუნველყოფის ძირითად ნაწილს წარმოადგენს ადამიანისა და კომპიუტრის ურთიერთობის კომპიუტერული ენები.

ინფორმაციული უზრუნველყოფის საფუძველს წარმოადგენს მონაცემები, რომლითაც სარგებლობენ დამპროექტებლები დაპროექტების დროს, საპროექტო გადაწყვეტილების მიღებისათვის. ეს მონაცემები შეიძლება წარმოდგენილი იყოს იმ დოკუმენტების სახით, რომლებიც შეიცავენ საცნიბარო ინფორმაციას. ინფორმაცია ინახება მონაცემთა ბაზებში, ანუ მონაცემთა ბაზის ფორმატის (mdb, db, exl და სხვა) ფაილების სახით. მონაცემთა ბაზების განაწილება/განლაგება შეიძლება იყოს ლოკალური ან ქსელური. ქსელური ბაზების უპირატესობა, ლოკალურ ბაზებთან შედარებით, იმაში გამოიხატება, რომ ასეტი ტიპის ბაზებთან მუშაობა, ანუ ამ ბაზებში არსებული ინფორმაციით სარგებლობა, ინფორმაციის მოდიფიცირება და ინფორმაციის დამატება, შეუძლია ქსელში ჩართულ რამოდენიმე მომხმარებელს ერთდროულად, რაც საგრძნობლად ზრდის მუშაუნარიანობას. მონაცემთა ბაზის ფაილებთან სამუშაოდ საჭიროა პროგრამული უზრუნველყოფის (აპლიკაციის არსებობა). ესეთი პროგრამული უზრუნველყოფების შექმნა ხდება ისეთი პროგრამირების ენების გამოყენებით როგორიცაა მაგალითად: C++, Pascal, Basic და სხვა. ზემოთ თქმულიდან გამომდინარე მონაცემთა შენახვის და მათთან მუშაობის არსებული სტრუქტურა შემდეგნაირად გამოიყურება:



ნახ.№2



ნახ.№3

ადს–ის არქიტექტურები

ადს–ის არქიტექტურების შეფასებისას ძიროთად კრიტერიუმებს წარმოადგენს ის, თუ რამდენად შესაძლებელია სისტემის მოდიფიცირება მომხმარებლის მიერ. ამ თვალსაზრისით განსაზღვრავენ შემდეგ სისტემებს:

1. ხისტი არქიტექტურის სისტემები, რომლებშიც ტექნიკური, პროგრამულ-მათემატიკური უზრუნველყოფა ხისტად არის განსაზღვრული და მომხმარებელს არა აქვს უფლება სისტემაში ცვლილების შეტანისა.

   ხისტი არქიტექტურის სისტემები შედარებით ადვილი დასამზადებელია, მაგრამ მნიშვნელოვანი პრობლემები იქმნება სისტემის ადაპტაციისას მომხმარებლის კონკრეტულ ამოცანაზე. ამას იწვევს ის ფაქტი, რომ ხისტი არქიტექტურის სისტემებში მომხმარებელს, კონკრეტული ამოცანის გათვალისწინებით, შეუძლია სისტემის მოდიფიცირება მხოლოდ ინფორმაციული უზრუნველყოფის დონეზე.

2. ღია არქიტექტურის სისტემები, რომლებიც საშუალებას იძლევიან მომხმარებლის მხრიდან განხორციელდეს სისტემის მოდიფიცირება, როგორც ტექნიკურ, ასევე პროგრამულ-მათემატიკური უზრუნველყოფის დონეზე.

   ღია არქიტექტურის სისტემები შედგებიან ცალკეული ტექნიკური და პროგრამული მოდელებისაგან, მომხმარებელს შესაძლებლობა აქვს ცალკეული მოდულების შერჩევით განსაზღვროს ადს–ის არქიტექტურა. მოდულების მიმართ ამ შემთხვევაში წამოყენებულია ორი ძირითადი მოთხოვნა:

   1) უნიფიცირება და 2) ინფორმაციული თავსებადობა. ტექნიკური მოდულების

ინფორმაციული თავსებადობისათვის გამოიყენება სპეციალური მოწყობილობები, რომლებსაც “ადაბტორები” ეწოდება, ხოლო პროგრამული მოდულების ინფორმაციული თავსებადობისათვის გამოიყენება პროგრამები, რომელთაც “დრაივერები” ეწოდებათ.

2.1. ექსპერტული სისტემები თავისი შინაარსით განეკუთვნებიან ღია არქიტექტურის სისტემებს, რომლებიც საშუალებას იძლევიან განხორციელდეს ადაპტაცია მომხმარებლის ამოცანების მათემატიკური გადაწყვეტილების დონეზე. ექსპერტული სისტემის დამახასიათებელი თვისებები შეიძლება შემდეგნაირად ჩამოვაყალიბოთ:

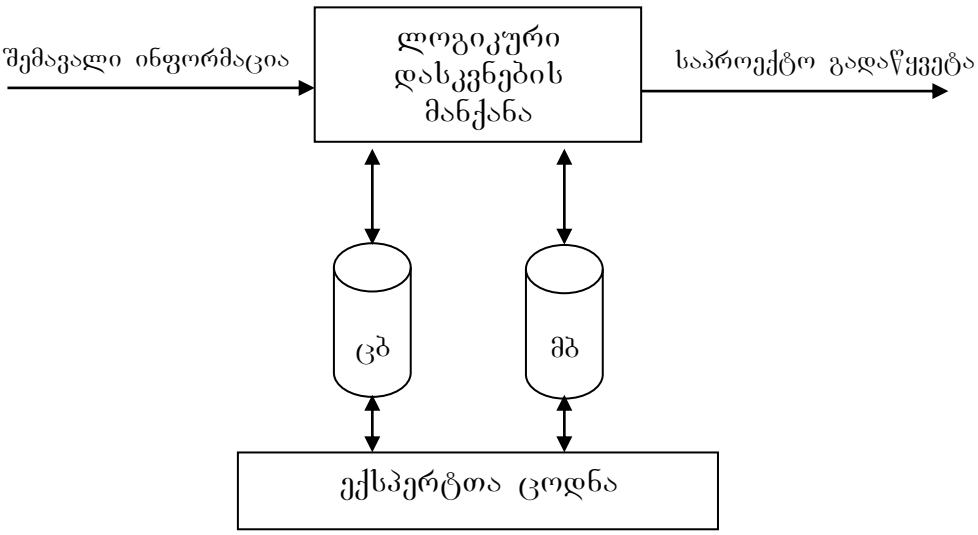
- ექსპერტული სისტემა შემოსაზღვრულია რაღაც საპრობლემო არით.
- ექსპერტულ სისტემას შეუძლია გადაწყვეტილების მიღება არაზუსტი ან საეჭვო მონაცემების შემთხვევაში
- მას შეუძლია ახსნას თავისი გადაწყვეტილების მიღების თანმიმდევრობა.
- ექსპერტულ სისტემაში ფაქტები და გადაწყვეტილების მიღების მექანიზმი მკვეთრად გამიჯნულია ერთმანეთისგან.
- ექსპერტულ სისტემას საფუძვლად უდევს წესების გამოყენება.
- საბოლოო შედეგი არის არა რაღაც ციფრები, ფუნქციები და გრაფიკები, არამედ ადამიანისათვის გასაგებ ენაზე ჩამოყალიბებული წინადადება. ანუ მიღებული გადაწყვეტილება.

ექსპერტულ სისტემებში გვაქვს ლოგიკური დასკვნების მანქანა , რომლის საშუალებითაც ხდება საპროექტო გადაწყვეტილების მიღება ცოდნის ბაზისა და მონაცემთა ბაზაში არსებული მონაცემების დახმარებით. ლოგიკური დასკვნების მანქანა გადაწყვეტილების მიღებას ახორციელებს ორი ძირითადი მეთოდით: “გადაწყვეტილების მიღების პიურდაპირი ჯაჭვი” და

“გადაწყვეტილების მიღების უკუ ჯაჭვი”. “გადაწყვეტილების მიღების პიურდაპირი ჯაჭვი”–ს დროს ჩვენ, არსებული ფაქტების საფუძველზე, მივდივართ დასკვნამდე. “გადაწყვეტილების მიღების უკუ ჯაჭვი”–ის დროს, უკვე არსებულ ჰიპოტეზას მივყავართ ამ ჰიპოტეზის დამადასტურებელ ფაქტებამდე.

ცოდნის ბაზა შედგება წესებისა და ფაქტების ერთობლიობისგან. ფაქტები წარმოადგენს მცირევადიან ინფორმაციას, ვინაიდან ისინი შეიძლება შეიცვალონ კონსულტაციის დროს. წესები უფრო გრძელვადიანი ინფორმაციაა იმ გაგებით, რომ წესებმა შეიძლება შეცვალონ არსებული წესები. რით განსხვავდება ასეთი მიდგომა მონაცემთა ბაზების მეთოდიკიდან: მონაცემთა ბაზებში ფაქტები საკმაოდ პასიური სიდიდეა, ანუ ფაქტები ან არის და ან არ არის. ცოდნის ბაზაში კი წესები აქტიურად ცდილობენ შეავსონ ინფორმაციული უკმარისობა. ცოდნის ბაზაში ინახება ინფორმაცია/მონაცემები რომლებიც მიღებულია ექსპერტების მიერ ევრისტიკული მეთოდებით. .მონაცემთა ბაზაში ინახება მონაცემები, რომლითაც სარგებლობენ დამპროექტებლები დაპროექტების დროს, საპროექტო გადაწყვეტილების მოღებისათვის. ცოდნის ბაზისა და მონაცემთა ბაზის შევსება/მოდიფიცირება ხდება ექსპერტების მიერ, ექსპერტთა ცოდნისა და გამოცდილების გათვალისწინებით. ექსპერტისგან ცოდნის ამოღება არის ერთერთი ყველაზე რთული ამოცანა. იდეალურ ვარიანტში შეგვიძლია გამოვყოთ რაიმე საპრობლემო არე და ექსპერტსა და პროგრამისტს საშუალება მივცეთ ურთიერთშეთანხმებულად იმუშაონ ამ პრობლემის გადაწყვეტაზე. მაგრამ ეს საკმაოდ ძვირადღირებილი პროცესია, ამიტომ აუცილებელი ხდება ცოდნის ამოღების პროცესის ავტომატიზაცია.

ექსპერტული სისტემები გრაფიკულად ასე გამოიყურება:



ნახ.№4 ექსპერტული სისტემა

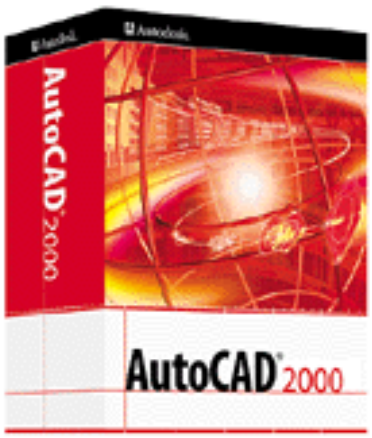
2.2 <u>ობიექტზე ორიენტირებული არქიტექტურა.</u> ასეთი არქიტექტურის დროს განასხვავებენ მშობელ და შვილ სისტემებს, სადაც შვილი სისტემისთვის ხელმისაწვდომა მშობლის ყველა რესურსი. მომხმარებლის მხრიდან შესაძლებელი ხდება სისტემის მოდიფიცირება პროგრამულ-მათემეტიკური უზრუნველყოფის დონეზე ანუ დამროექტებელს საშუალება აქვს საკუთარი შეხედულებისამებრ შექმნას სისტემის პროგრამულ-მათემატიკური უზრუნველყოფა. მომხმარებლისათვის, საკუთარი პროგრამულ-მათემატიკური უზრუნველყოფის შექმნის დროს, ხელმისაწვდომია მშობელი სისტემის ყველა რესურსი (ფუნქციები, კლასები, ობიექტები). და მათი გამოყენებით მომხმარებელს სისტემის მოდიფიცირების საშუალება ეძლევა სისტემისკონკრეტულ ამოცანაზე ადაპტაციისათვის. "შვილი სისტემის" (მოდულის) შექმნა "მშობელი სისტემის" რესურსების გამოყენებით

შესაძლებელი ხდება ამ სისტემაში არსებული დაპროგრამების ენების გამოყენებით. დამპროექტებელს ასევე საშუალება ეძლევა თავის მიერ შექმნილი და უკვე არსებული ცალკეული მოდულების შერჩევით განსაზღვროს "მშობელი სისტემის" არქიტექტურა. როგორც ძემოთ ავღნიშნეთ, ესეთი მოდულების მიმართ წამოყენებულია უნიფიცირებისა და ინფორმაციული თავსებადობის მოთხოვნა.

ადს–ის ობიექტზე ორიენტირებული არქიტექტურის ერთადერთ ნაკლად შეიძლება ჩაითვალოს "მშობელი სისტემის" აუცილებელი არსებობა, რაც იმას ნიშნავს, რომ დამპროექტებლის მიერ საკუთარი მოდულის შექმნა/დაპროექტება ხდება მშობელი სისტემის გარემოში და ამ მოდულის მშობელი სისტემის გარეშე მუშაობა შეუძლებელია.

ადს–ის ობიექტზე ორიენტირებული არქიტექტურის ერთერთ საუკეთესო მაგალითს წარმოადგენს AutoCAD–ი.

AutoCAD–ის მიმოხილვა.



AutoCAD–ი უნივერსალური გრაფიკული სისტემაა, რომელიც განკუთვნილია ნებისმიერი სპეციალისტისთვის რომელიც მუშაობს ტექნიკურ თუ საპრეზენტაციო გრაფიკასთან. სახატავი გრაფიკული რედაქტორებისგან განსხვავებით AutoCAD–ი მუშაობს არა რასტრულ არამედ ვექტორულ გამოსახულებასთან. მონიტორის ეკრანზე არსებული გამოსახულების დათვალიერებისას, სინამდვილეში ჩვენ ვხედავთ ფერადი წერტილების დიდ

რაოდენობას, რომლებიც ერთადაა თავმოყრილი და ერთიანობაში ქმნიან არსებულ გამოსახულებას. აქედან გამომდინარე ნებისმიერი გრაფიკული ფაილი უნდა შეიცავდეს ინფორმაციას იმის შესახებ თუ როგორ გამოისახოს ამ წერტილების ერთობლიობა მონიტორის ეკრანზე. ამის მიღწევა ხდება არსებული წერტილების თვისებების მათემატიკური აღწერით და ამ აღწერის გრაფიკულ ფაილში დამახსოვრებით. გრაფიკული გამოსახულების მათემატიკური აღწერის სხვადასხვა მეთოდების გამო არსებობს გრაფიკული გამოსახულების მრავალი ფორმატი, თუმცა ყველა ეს ფორმატი შეიძლება მივაკუთნოთ ორ ძირითად მოდელს: რასტრული და ვექტორული. რასტრული მოდელი წარმოადგენს ცალკე აღებული სხვადასხვა ფერის წერტილების ერთობლიობას, რომლებიც ქმნიან საერთო სურათს. რასტრული მოდელის საუკეთესო მაგალითს წარმოადგენს დასკანირებული გამოსახულება ან Adobe Photoshop ან Windows Paint–ში შექმნილი გამოსახულება. რასტრული გრაფიკის გამოყენება მაღალი ხარისხის ფოტოგრაფიული გამოსახულების მიღების საშუალებას იძლევა. უარყოფიდ მხარედ შეიძლება ჩაითვალოს ფაილის მოცულობა, რედაქტირების სირთულე, რაც იმას გულისხმობს, რომ რასტრული გამოსახულების რედაქტირებისას, იმ შემთხვევაშიც კი თუ ეს გამოსახულება უბრალო ხაზია ან რკალი, ხდება ყველა პიქსელის რედაქტირება. რასტრული გამოსახულების გადიდების ან დაპატარავების დროს, გამოსახულების ხარისხი საგრძნობლად უარესდება. რასტრული მოდელისგან განსხვავებით, ვექტორული მოდელი წარმოადგენს ხაზების/ვექტორების ერთობლიობას, რომლებიც ქმნიან საერთო გამოსახულებას. ამ შემთხვევაში ფაილში ინახება ინფორმაცია არა ცალკეული წეწრტილების შესახებ, არამედ ინფორმაცია იმ ელემენტების (ვექტორების) შესახებ რომლისგანაც შედგება მთლიანი გამოსახულება. ასეთი გამოსახულება უფრო კომპაქტურია და ადვილად რედაქტირებადი, ვინაიდან რედაქტირების დროს ხდება

ცალკეული ვექტორების რედაქტირება სხვებისგან დამოუკიდებლად. ვექტორული გამოსახულების ზომების შეცვლა არ იწვევს ხარისხის გაფუჭებას და შემავალი ობიექტების საერთო განლაგების დარღვევას. როგორც ზემოთ ავღნიშნეთ AutoCAD–ი მუშაობს ობიექტების გეომეტრიულ აღწერასთან, რაც ადს–ის ამოცანებით არის განპირობებული. მაგალითად წრფე AutoCAD–ში აღიწერება ორი წერტილით, წრე აღიწერება ცენტრით და რადიუსით და ასე შემდეგ. ობიექტის წარმოდგენა მისი გეომეტრიული პარამეტრების აღწერის მეთოდით უფრო კომპაქტურია, რაც მთავარია გეომეტრიული მოდიფიცირების საშუალებას იძლევა (რაც თავისთავად ყოველთვის საჭიროა საინჟინრო პროექტირების დროს), და ობიექტის წარმოდგენა მისი გეომეტრიული პარამეტრების აღწერის მეთოდით გამოიყენება წარმოების ტექნოლოგიური მომზადების ავტომატიზებულ სისტემებში.

AutoCAD–ში მუშაობა ხდება ინტერაქტიულ რეჟიმში, რაც საშუალებას აძლევს მომხმარებელს დინამიურად, დროის არსებულ მომენტში აკონტროლოს თავისი ურთიერთობა სისტემასთან. AutoCAD–ს გააჩნია ბრძანებათა მოქნილი სისტემა, რაც საშუალებას იძლევა შესრულდეს ბრძანებები ნებისმიერი თანმიმდევრობით. AutoCAD–ში ბრძანების შეყვანა ხდება კლავიატურაზე აკრეფით ანუ ბრძანებათა ხაზის (Command Line) მეშვეობით, მენიუდან არჩევით ან ინსტრუმენტების პანელზე არსებულ ღილაკებზე დაჭერით.

AutoCAD–ში არსებული ბრძანებები შეიძლება დავყოთ სამ კატეგორიად:

- **ხაზვის ბრძანებები**
- **რედაქტირების ბრძანებები**
- **ფუნქციონალური ბრძანებები**

ხაზვის ბრძანებებებს მიეკუთვნება ისეთი ბრძანებები, რომლებიც მომხმარებელს AutoCAD–ის სისტემაში ნებისმიერი პრიმიტივის დახაზვის საშუალებას აძლევენ. ასეთ ბრძანებებს მიეკუტვნება, მაგალითად ბრძანებები: Line, Circle, Arc და სხვა.

რედაქტირების ბრძანებების სასუალებით მომხმარებელს შეუძლია უკვე დახაზული ობიექტების გეომეტრიული თუ სხვა თვისებების შეცვლა/რედაქტირება. მაგალითად Stretch, Lengthen, Extrude და სხვა.

ფუნქციონალური ბრძანებები მომხმარებელს საშუალებას აძლევენ შეცვალოს სისტემის მიმდინარე პარამეტრები, ისეთები როგორიცაა: SNAP (კურსორის გადაადგილების ბიჯის დაყენება), POLAR (დამმხმარე, პოლარულ წრფეების ჩართვა/გამორთვა), GRID (ბადის გამოჩენა/გაქრობა), ORTHO (ორთოგონალური ხაზების გავლება), OSNAP (ობიექტების მიბმის ავტომატური კონტროლი), OTRACK (დამმხმარე, ობიექტების ურთიერთგანლაგების, წრფეების ჩართვა/გამორთვა), LWT (ხაზების სისქის გამოსახვის ჩართვა/გამორთვა) და სხვა.

AutoCAD–ში ყველა გრაფიკული ობიექტი ფორმირდება სხვადასხვა ტიპის ელემენტებისაგან, რომლებიც ერთმანეთისგან განსხვავდებიან გეომეტრიული აღწერით. ასეთ ელემენტებს გრაფიკული პრიმიტივები ეწოდებათ. პრიმიტივებს გააჩნიათ კონკრეტული გეომეტრიული პარამეტრები (მაგალითად; საწყისი და საბოლოო წერტილის კოორდინატები, სიგრძე, რადიუსი და სხვა) და კონკრეტული თვისებები (მაგალითად: ფერი, ხაზის ტიპი, რომელიმე ფენისადმი (Layer) მიკუთვნება და სხვა).

AutoCAD–ი შეიცავს შემდეგ გეომეტრიულ პრიმიტივებს: წერტილი (Point), მონაკვეთი (Line), სხივი (Ray), ორმაგი ხაზი (MLine), ფიგურა (Solid), რკალი (Arc), ელიფსი (Ellipse), წრეწირი (Circle), მრუდი (Spline), პოლილაინი (Poliline), 3D პოლილაინი (3D Poliline), რეგიონი (Region), ბლოკი (Block), ატრიბუტი (Attdef), დაშტრიხვა (Hatch), ზომა (Dimension), ტექსტი (Text, Mtext), სამგანზომილებიანი ტეხილი (3D Face), ბადე (3D Mesh), სხეული (3D Solid).

ამ პრომიტივებს გააჩნიათ გარკვეული თვისებები, რომლებიც შეიძლება დავყოთ საერთო ხასიათის და სპეციფიკურ თვისებებად. საერთო ხასიათის თვისებებს მიეკუთვნება ისეთი თვისებები, რომლებიც დამახასიათებელია ყველა პრიმიტივისათვის, მაგალითად: პრომიტივის ფერი, რომელიმე Layer–ისადმი მიკუთვნება და სხვა. სპეციალურ თვისებებს მიეკუთვნება ისეთი თვისებები, რომლებიც დამოკიდებულია პრიმიტივის ტიპზე, მაგალითად "ხაზის ტიპი". იმის გასაგებად თუ რა თვისებები გააჩნია ამა თუ იმ ობიექტს, და რაც მთავარია ამ თვისებების მოდიფიცირებისათვის, გამოიყენება დიალოგური ფანჯარა "Object Properties". [2]

განვიხილოთ ის პროცედურები და ფუნქციები, რომლებიც გააჩნია AutoCAD–ს:

- **რამოდენიმე ნახაზის ერთდროულად გახსნის საშუალება.**

AutoCAD–ი, Windows_ის სტანდარტული ფუნქციების (drag&drop, Shift'ით ან Ctrl'ით არჩევა) გამოყენებით, ერთდროულად რამოდენიმე ნახაზთან მუშაობის საშუალებას იძლევა.

- **ობიექტების გადატანა / კოპირება.**

  ეს ფუნქცია ობიექტების კოპირების და/ან გადატანის საშუალებას იძლევა, ნახაზის შიგნით ან ნახაზიდან ნახაზში.

- **თვისებების კოპირება.**

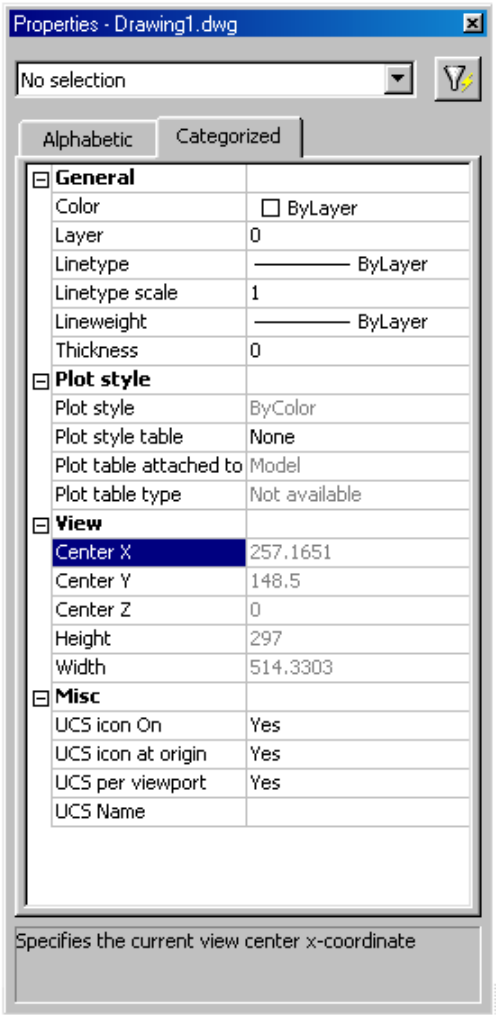
  AutoCAD–ის გარემოში მომხმარებელს AutoCAD–ის ობიექტების თვისებების (ფერი, ფენისადმი მიკუთვნება, ხაზის ტიპი და სხვა) კოპირების საშუალება აქვს ნახაზის შიგნით ან ნახაზიდან ნახაზში.

- **გააქტიურებული ბრძანებიდან გამოუსვლელად ნახაზებს შორის გადასვლა.**

  ეს ფუნქცია მომხმარებელს, წინასწარ გახსნილ რამოდენიმე ნახაზში, აქტიური ბრძანების გაუქმების გარეშე, ნავიგაციის საშუალებას აძლევს.

- **ობიექტების თვისებების დიალოგური ფანჯარა.**

AutoCAD_ს გააჩნია ობიექტების თვისებების დიალოგური ფანჯარა, რომელიც აადვილებს ობიექტების თვისებების დათვალიერებისა და რედაქტირების პროცესს. ეს დიალოგური ფანჯარა აერთიანებს ისეთი ბრძანებების ფუნქციონალურ შესაძლებლობებს, როგორიცაა: DDMODIFY, DDEDIT, DDCHPROP და სხვა რედაქტირების სპეციფიკურ ბრძანებებს.



ობიექტების თვისებების
დიალოგური ფანჯარა.

ობიექტების თვისებების დიალოგური ფანჯარას გააჩნია თავისი ფუნქციები :

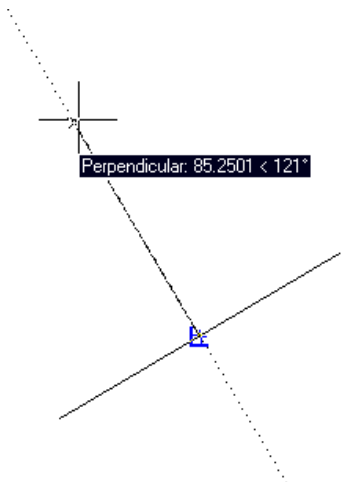
ტიპის მიხედვით რედაქტირება. ეს ფუნქცია ამორჩეული ობიექტების ნაკრებს ფილტრავს ტიპების მიხედვით, რაც საშუალებს აძლევს მომხმარებელს ამორჩეული ობიექტებიდან თითოეულს ცალცალკე შეუცვალოს თვისება.

რედაქტირება ამორჩეული ობიექტის გარეშე. როდესაც არ არის შერჩეული ობიექტი, დიალოგურ ფანჯარაში ჩანს ნახაზის ისეთი მიმდინარე თვისებები როგორიცაა: ინფორმაცია UCS–ის შესახებ, ჰიპერლინკები, ბეჭდაზე გამოტანის სტილები და სხვა.

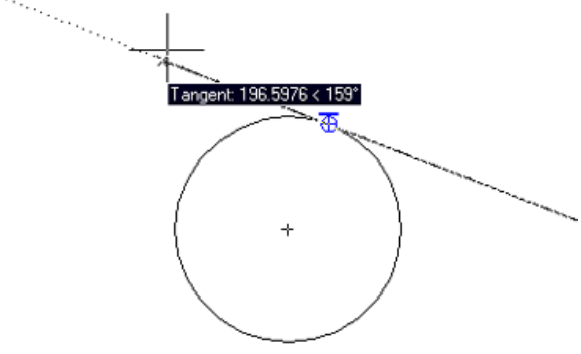
დიალოგური ფანჯრის სანიშნები. დიალოგური ფანჯრის სანიშნები საშუალებას იძლევიან დავათვალიეროთ ობიექტების თვისებები კატეგორიების მიხედვით (მაგალითად: საერთო თვისებები, გეომეტრიული თვისებები და სხვა) ან ალფავიტის მიხედვით დაწყობილი თანმიმდევრობით.

- **ავტომატური მიბმა და ავტოტრასირება.**

AutoCAD–ის ეს ხაზვის პროცედურა საგრძნობლად აჩქარებს ხაზვის დროს ობიექტის ზუსტი კოორდინატების შეყვანას ნახაზის სხვა ობიექტების მიმართ ისეთი დამოკიდებულებების გამოყენების მეთოდით როგორებიცაა: პარალელურობა, პერპენდიკულარულობა, მხები და სხვა.



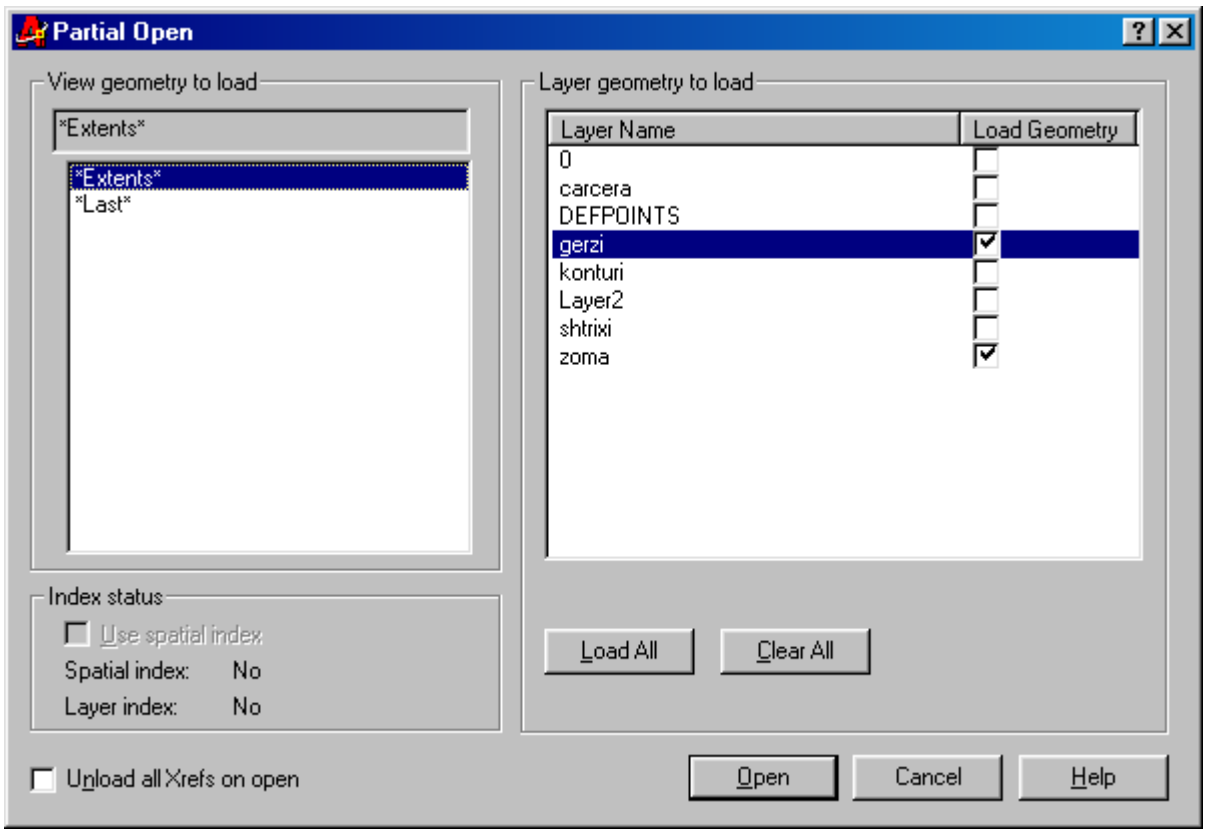
პერპენდიკულარობის მაჩვენებელი დამმხმარე სხივი



მხების მაჩვენებელი დამმხმარე სხივი

- **ფაილის ნაწილობრივი ჩატვირთვა.**

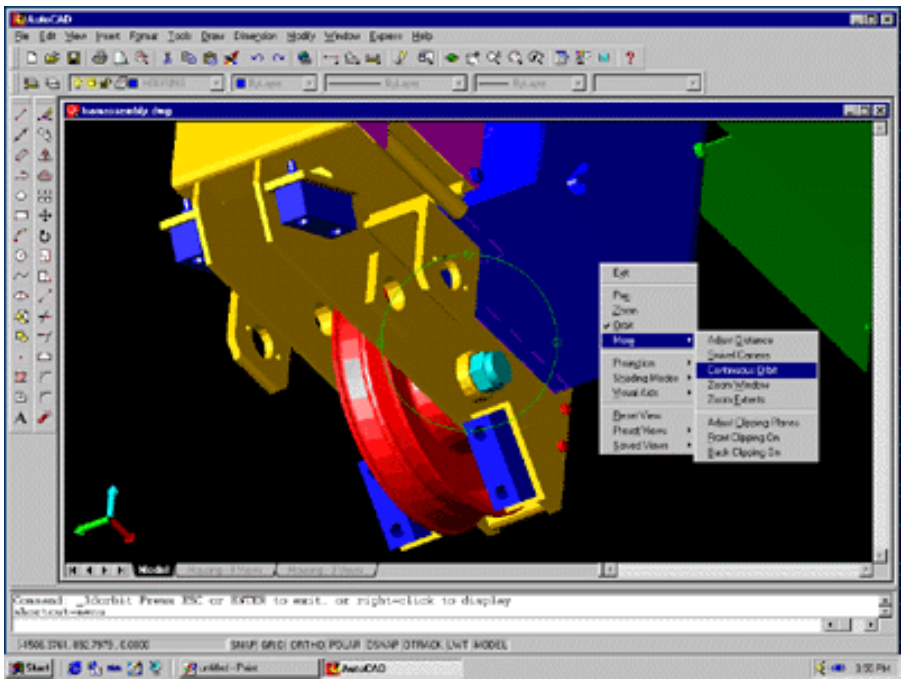
დიდი მოცულობის ფაილებთან მუშაობის დროს ხდება მუშაობის ტემპის საგრძნობი შემცირება. აქედან გამომდინარე, მომხმარებელს შეუძლია ჩატვირთოს არსებული ნახაზის მხოლოდ მისთვის სასურველი ფენები (Layer) ან ხედვის არეები. საჭიროების დროს კი შესაძლებელია სხვა (უკვე საჭირო) ფენების ან ხედვის არეების დამატებითი ჩატვირთვა.



ფაილის ნაწილობრივი ჩატვირთვის
ფანჯარა

- **3D Orbit**

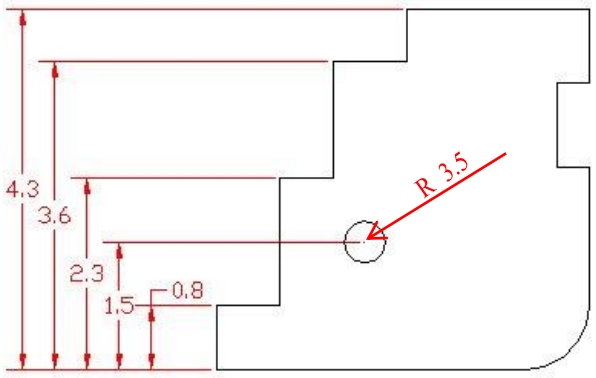
ეს არის AutoCAD–ის პროცედურა რომელიც დამყარებულია ACIS Kernal Geometrical Processor–ზე და მომხმარებელს საშუალებას აძლევს შექმნას სამგანზომილებიანი ობიექტები სამგანზომილებიან სივრცეში. მომხმარებელს შეუძლია შექმნას სამგანზომილებიანი ობიექტი წიბოების, ზედაპირების, მყარი ტანის სხეულებისგან და 3D Orbit–ის საშუალებით შეასრულოს ისეთი ბრძანებები როგორიცაა: ობიექტის შემობრუნება, გადაადგილება, ამ სამგანზომილებიანი ობიექტის რედაქტირება და სხვა.



სამგანზომილებიანი ობიექტი 3D Orbit
რეჟიმში დათვალიერებისას

- **ზომების ავტომატური გამოთვლა/გამოტანა.**

ობიექტის გეომეტრიული ზომების გამოთვლა დამპროექტებლისათვის საკმაოდ საჭირო და რაც მთავარია შრომატევადი პროცესია. AutoCAD–ის ზომების ავტომატური გამოთვლის/გამოტანის ფუნქცია საგრძნობლად ზრდის შრომისუნარიანობას, ვინაიდან ის დამპროექტებელს ადვილად, ობიექტის გეომეტრიის მითითებით, ამ ობიექტების ზომების ავტომატურ გამოთვლა /გამოტანის საშუალებას აძლევს. AutoCAD–ის ეს ფუნქცია საშუალებას იძლევა გამოვთვალოთ და ეკრანზე გამოვიტანოთ ნებისმიერი ტიპის ზომა, მაგალითად: სიგრძე, რადიუსი, კუთხის ზომა და სხვა.



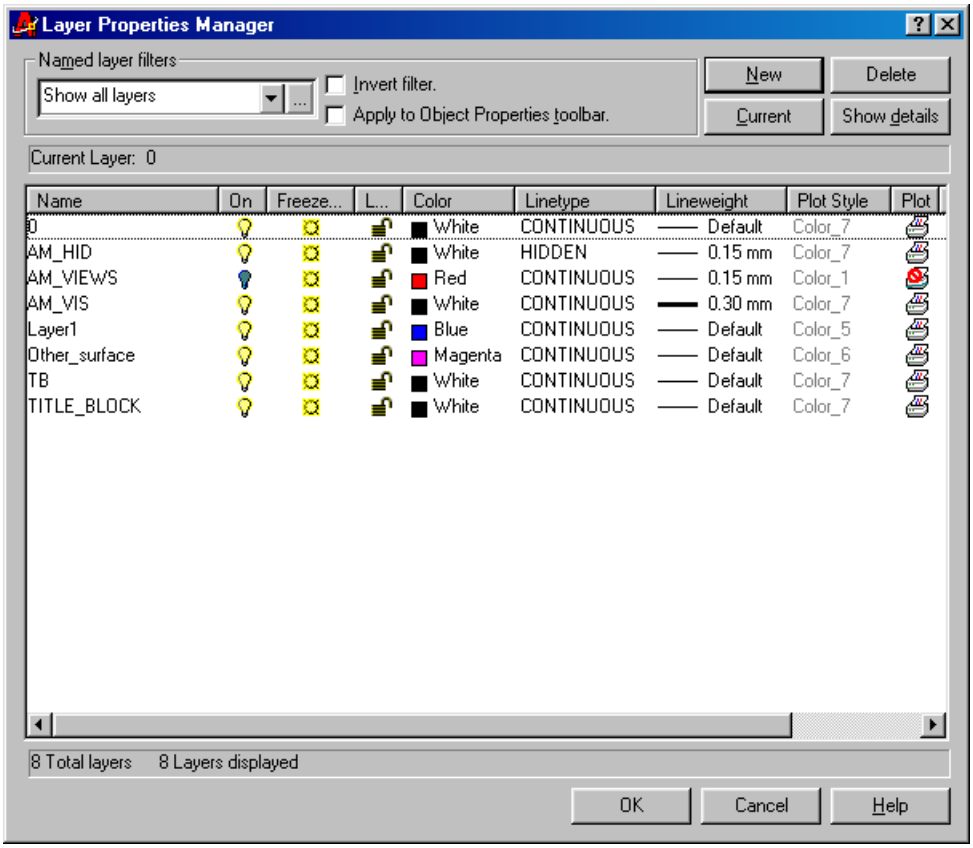
ზომების გამოტანა

ამ ფუნქციის მეშვეობით დამპროექტებელს შეუძლია შემდეგი საშუალებების გამოყენება:

- ზომების სტილების შერჩევის გაადვილებული საშუალება.
- AutoCAD DesignCenter_ის საშუალებით ზომების ახალი სტილების იმპორტი ნახაზში.
- ზომების სტილების სწრაფი და დინამიური რედაქტირების საშუალება.

- **ფენების (Layer) მენეჯერი**

ფენების მენეჯერი გამოიყენება AutoCAD_ში შექმნილი ნახაზის ფენების შესაქმნელად, დასათვალიერებლად, ამ ფენების და მათი თვისებების რედაქტირებისათვის.



ფენების მენეჯერის ფანჯარა

ფენების მენეჯერი შეიცავს:

- 255 სომბოლოიანი (პრობელებისა და სხვა არაალფავიტური სიმბოლოების) ფენების სახელების გამოყენების საშუალება.
- ფენის სახელებში ზედა და ქვედა რეგისტრის სიმბოლოების გამოყენების საშუალება.
- ფენათა ფილტრი.
- Lineweight, Plot/No plot, Freezing/Thawing Viewports Plot Style ატრიბუტებისათვის ფენების გამოყენების საშუალება.

- **ტექსტური ფორმატები.**

AutoCAD_ის მომხმარებელს ტექსტურ ინფორმაციასთან მუშაობის საშუალებაც აქვს, ვინაიდან AutoCAD_ს სხვადასხვა ტექსტური ფორმატების გამოყენების საშუალება გააჩნია. ტექსტური ფორმატები შეიცავენ შემდეგ თვისებებსა და საშუალებებს:

- Mtext რედაგტორი ინტერვალებისა და არაალფავიტური სიმბოლოების გამოყენების საშუალებას იძლევა.
- ტექსტის გასწორების საშუალება (top, middle, and bottom)
- ტექსტის ზომების მოდიფიცირება.
- ზედა და ქვედა რეგისტრის სიმბოლოების დამნოყენების საშუალება.
- windows_ის სხვადასხვა ფონტების გამოყენების საშუალება.

- **პირდაპირი კავშირი ვებ-ბროუზერთან**

AutoCAD_ის მომხმარებლებს ხშირად სჭირდებათ ინტერნეტი AutoCAD_ის კომპონენტების, მზა ობიექტების, ბლოკების მოსაძებნად მათი შემდგომში AutoCAD_ის ბაზაში ჩამატების მიზნით. აქედან გამომდინარე AutoCAD–ში ინტეგრირებულია ისეთი საშუალებები რომლებიც აადვილებს ინტერნეტში არსებული ინფორმაციის მოძიებას და მათ გამოყენებას AutoCAD–ის გარემოდან გამოუსვლელად.

ბროუზერის ფანჯარა

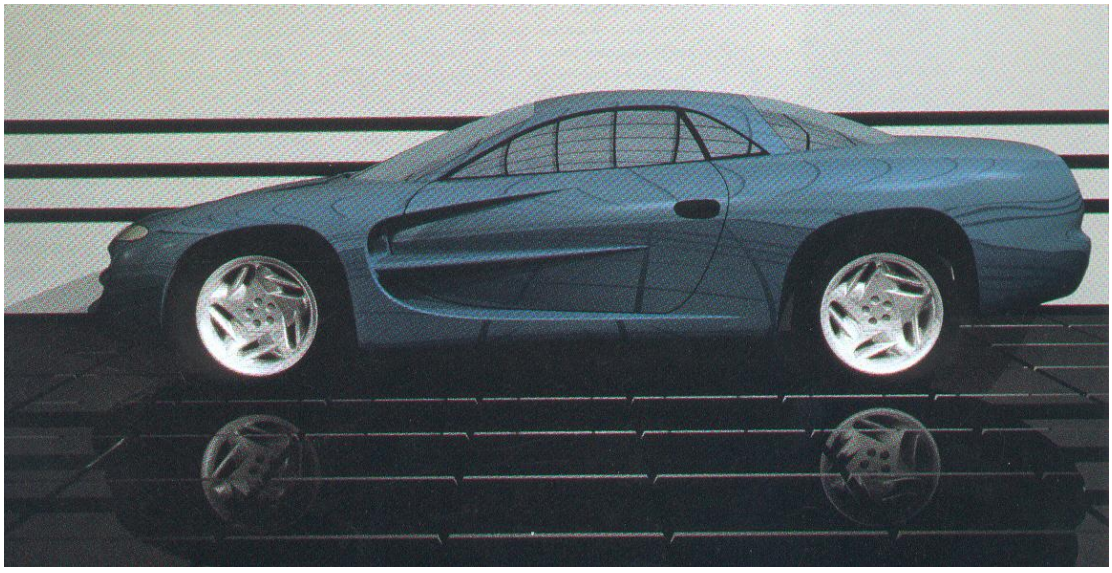
- **Render**

AutoCAD_ის მომხმარებელს Render_ის მეშვეობით შეუძლია კარკასული და მყარი ტანის სხეულებისგან აგებული ობიექტის/ფიგურის ფოტორეალისტური სურათის შექმნა, რაც გულისხმობს ჩრდილების, განათებების, ტექსტურების, გამჭირვალობის, ფონური სურათების გამოყენებას. AutoCAD–ში Render_ის სამი ტიპი არსებობს:

Simple Render. AutoCAD_ის სტანდარტული რენდერი.

Photo Real Render. რენდერი პარამეტრიზებული გამჭირვალობის და ჩრდილების გამოყენებით.

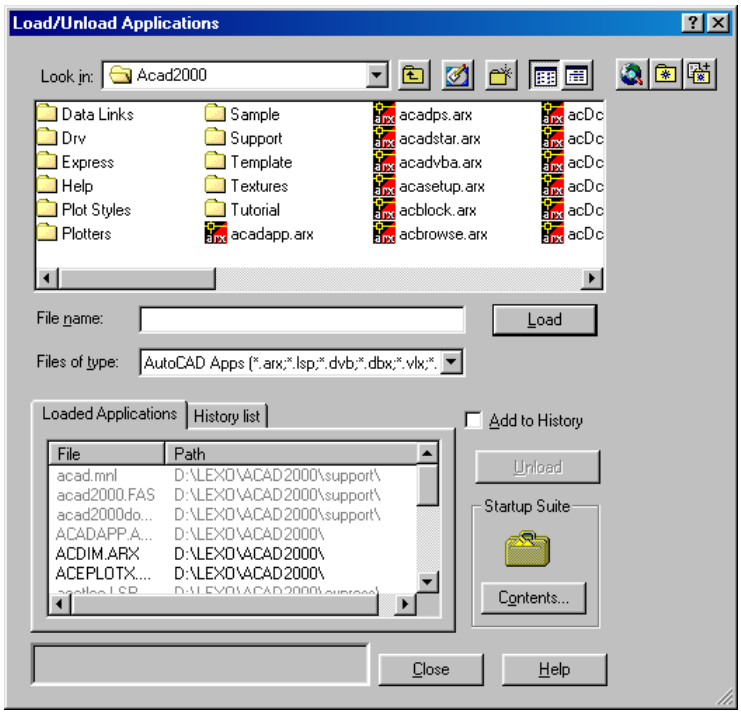
Photo Raytrace Render. ფოტორეალისტური სხივური რენდერი, რომელიც იყენებს სხივური დაყოფის მეთოდს, უფრო რეალისტური შუქჩრდილების მიღების მიზნით.



რენდერით მიღებული, AutoCAD_ში
დახაზული მანქანის მოდელის
სურათი

- **დიალოგური ფანჯარა Appload.**

დიალოგური ფანჯარა Appload, Autocad_ის მომხმარებელს ხშირად ხმარებადი სამომხმარებლო პროგრამების და მომხმარებლის მიერ Visual Basic, Visual LISP, და ObjectARX_ზე დაწერილი მოდულების სიის შექმნის, შენახვის და რედაქტირების საშუალებას აძლევს, მათი ავტომატურად ჩართვის მიზნით.

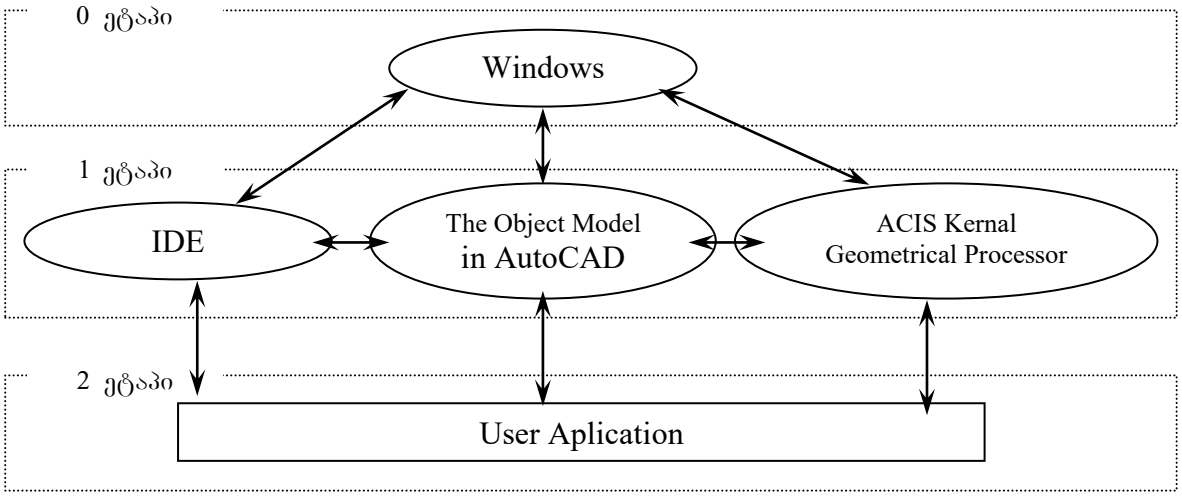


დამპროექტებლის AutoCad–ის გარემოში მუშაობის შედეგად მიიღება ფაილი, რომელიც შეიცავს გეომეტრიულ და დამმხმარე ინფორმაციას, რომელიც სრულიად აღწერს შექმნილ გრაფიკულ ობიექტს. დამპროექტებელს საშუალება აქვს ვიზუალიზაციისა და რენდერის საშუალებით, შეაფასოს შექმნილი ობიექტის გეომეტრიული პარამეტრები და პლოტერი ან პრინტერის საშუალებით შექმნას ობიექტის ნახაზი და დოკუმენტაცია ქაღალდზე.

AutoCad–ში შექმნილი პროექტის ფაილი ინახება ვინჩესტერზე, დისკეტებზე ან კომპაკტდისკებზე და სურვილის შემთხვევაში დამპროექტებელს შეუძლია მისი გახსნა რედაქტირებისათვის, კოპიის ან პროტოტიპის შექმნისათვის, ან ამობეჭდისათვის, რაც საგრძნობლად ამცირებს დაპროექტების შრომატევადობას.

AutoCad–ის სისტემის დამპროექტებლებმა, გაითვალისწინეს რა მომხმარებელთა საკმაოდ ფარტო წრის მოთხოვნები, სისტემაში ჩადეს AutoCad–ის ადაბტაციის ფართო საშუალებები ნებისმიერ საპროექტო სფეროში. სისტემის ისეთ ადაპტაციას, რომელიც თავისთავად შეიცავს საკუთარი ბლოკების შექმნას, მენიუების შექმნას და მოდიფიცირებას, დიალოგური ფანჯრების შექმნას, საკუთარი ქვეპროგრამების შექმნას Basic, AutoLisp, C++ პროგრამირების ენების საშუალებით და მრავალ სხვას, სჭირდება გარკვეული დრო და ძალისხმევა. როგორც წესი ესეთი სამუშაო სრულდება, ამ საქმის მცოდნე სისტემის პროგრამისტის მიერ, რომელიც ამზადებს სისტემას, ახდენს მის მოდიფიცირებას, ატარებს სასწავლო ტრეინინგებს სპეციალისტებისთვის, რომლებმაც უნდა გამოიყენონ ეს სისტემა თავის სფეროში. სამუშაოს ესეთი ორგანიზება საშუალებას იძლევა სწრაფად და ეფექტურად გამოვიყენოთ AutoCad–ი პრაქტიკაში და არ მოხდეს სპეციალისტების მოცდენა მათი პირდაპირი პროფესიონალური მოვალეობებისგან. თუმცა არის ისეთი ოპერაციები და ფუნქციები, რომლებიც უნდა იცოდეს ყველამ ვინც აპირებს AutoCad–ის გამოყენებას. მაგალითად სისტემაში შესვლა, უკვე შექმნილი პროექტის გახსნა/ჩატვირთვა, ორგანზომილებიანი და სამგანზომილებიანი ობიექტების შექმნა და რედაქტირება, კომპლექსური ნახაზის შექმნა და მისი შენახვა, პლოტერისა და/ან პრინტერის საშუალებით ნახაზის ქაღალდზე ამობეჭდვა და სხვა.

ადაბტაციის სტრუქტურა შემდეგნაირად გამოიყურება:



ნახ.№4 ადაბტაციის სტრუქტურა

განვიხილოთ ამ სტრუქტურის შემადგენელი თითოეული ელემენტი უფრო დაწვრილებით.

გეომეტრიული პროცესორი არის ნებისმიერი სამგანზომილებიანი მოდელირების სისტემის ბირთვი და იგი თავისთავად წარმოადგენს CAD სისტემის ძირითადი მათემატიკური ფუნქციების ბიბლიოთეკას, რომელიც განსაზღვრავს და ინახავს 3D ფორმებს და მომხმარებლის ბრძანებების მოლოდინის რეჟიმში იმყოფება. გეომეტრიული პროცესორი ამუშავებს მომხმარებლის ბრძანებებს, ინახავს მიღებულ შედეგებს და ამ შედეგების მონიტორის ეკრანზე გამოტანას ახორციელებს.

ACIS Kernal Geometrical Processor (გეომეტრიული პროცესორი) წარმოადგენს ობიექტზე ორიენტირებულ C++ გეომეტრიულ ბიბლიოთეკას რომელიც შედგება 35–ზე მეტი DLL ფაილისაგან და შეიცავს კარკასულ სტრუქტურებს, ზედაპირებს და მყარი ტანის მოდელებს. ის პროგრამის დამპროექტებელთათვის, გეომეტრიული

ოპერაციების, რთული მოდელების კონსტრუირებისა და მანიპულაციის, და მთელი რიგი ლოგიკური ოპერაციების, ფართო არჩევანს იძლევა.

2000 წელს კორპორაცია "Dassault Systemes"–მა შეისყიდა ACIS მწარმოებელი კორპორაცია "Spatial Technology Co." რასაც მოჰყვა ACIS გეომეტრიული პროცესორის მნიშვნელოვანი გაუმჯობესება. გეომეტრიული პროცესორის ახალ ACIS 6.3 ვერსიაში დამატებული იქნა მთელი რიგი კომპონენტები და ფუნქციები. მათი ჩამონათვალი და მოკლე აღწერა მოყვანილია ქვემოთ:

- **ACIS დაგლუვება** – გამოიყენება მთელი რიგი, დაგლუვების, რთული ოპერაციების ჩასატარებლად. შეიცავს დაგლუვების ისეთ საშუალებებს როგორიცაა დაგლუვება მუდმივი რადიუსით, დაგლუვება ცვლადი რადიუსით, წრფივი დაგლუვება და სხვა.

- **ACIS ლოკალური ოპერაციები** – გამოიყენება წიბოების გეომეტრიის ლოკალური ცვლილებისათვის მოდელის ტოპოლოგიის მინიმალური ცვლილების გათვალისწინებით.

- **ACIS ხაზების დაფარვა** - გამოიყენება მყარი ტანის, ზედაპირული და კარკასული მოდელების დაფარული ხაზების განსაზღვრისა და დაფარვისათვის. რაც მოდელის რეალისტიკული გამოსახვის საშუალებას იძლევა.

- **ACIS გარსები (Оболочки)** – გამოიყენება გარსის სხეულის შესაქმნელად წიბოების გადატანის ან მყარი ტანის ფურცლების (ფირფიტების) გამოყენების გზით.

- **ACIS დეფორმაცია** – დეფორმაცია სრულდება მოდელის შემომსაზღვრელი ჩარჩოს საშუალებით. რომლის ზემოქმედებითაც ხდება მოდელის დინამიურად გაღუნვა/დეფორმაცია.

- **ACIS გაფართოებული ზედაპირული მოდელირება** – გამოიყენება ზედაპირების მოდელირებისათვის მრუდების ერთობლიობის, ზედაპირის გაჭიმვის, ლოფტინგის და ბადური ზედაპირების გამოყენების მეთოდებით.

- **ACIS უჯრედული ტოპოლოგია** – გამოიყენება მოდელის პატარპატარა ქვეუჯრედებად დეკომპოზიციისათვის, რაც ისეთი უნიკალური ინფორმაციის მოპოვების საშუალებს იძლევა როგორიცაა მასალების თვისებები ან მოდელში ცვლილებები.

.

Integrated Development Environment (IDE)

კომპანია "**Microsoft**"–ის მიერ წარმოებული ინტეგრირებული დაპროექტების სისტემა **Visual C++** წარმოადგენს პროფესიონალურ პროგრამულ საშუალებას, რომლის მეშვეობითაც შესაძლებელია მომხმარებლის აპლიკაციების/პროექტების შექმნა. ეს აპლიკაციები შეიძლება წარმოდგენილი იყოს როგორც ცალკეული **Windows** აპლიკაციების/პროგრამების სახით, ასევე **AutoCAD**–ში ინტეგრირებული აპლიკაცია/მოდულების სახით. **Visual C++** ის გარემოში პროექტის შექმნა (დაპროგრამება) ხდება **C++** ობიექტზე ორიენტირებული დაპროგრამების ენის საშუალებით და **MFC (Microsoft foundation Classes)** კლასების ბიბლიოთეკის გამოყენებით. ვერსიების

გაუმჯობესებასთან ერთად **Visual C++** –ს მრავალი ახალი ფუნქცია და საშუალება ემატება. ქვემოთ მოყვანილია **Visual C++** –ის ძირითადი ფუნქციები და საშუალებები:

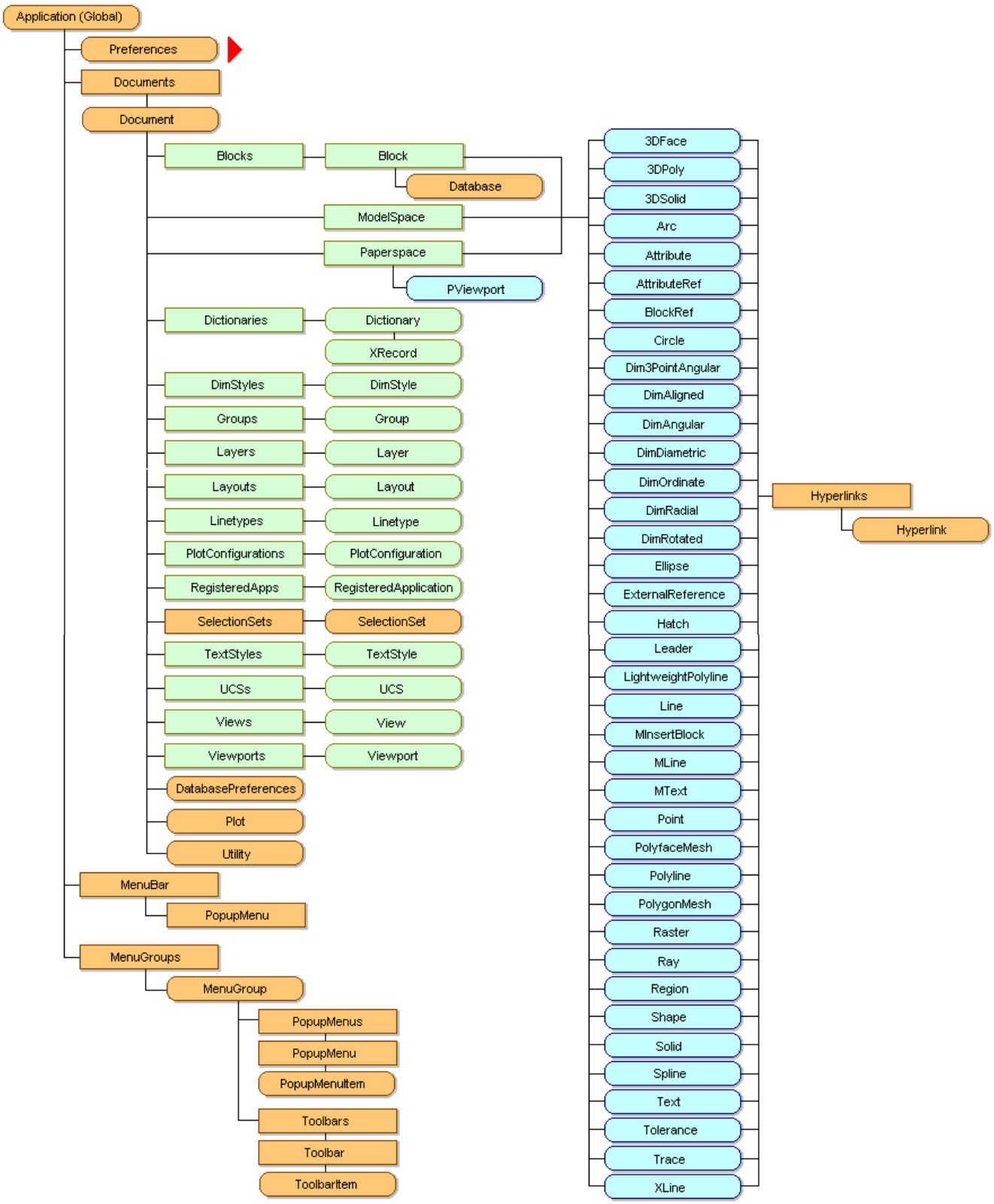
- ობიექტზე ორიენტირებული პროგრამირების ენის **C++** –ის ფუნქციებისა და კლასების გამოყენების შესაძლებლობა.
- **MFC (Microsoft foundation Classes)** კლასების ბიბლიოთეკა.
- **Active Template Library (ATL)** ბიბლიოთეკა.
- **Internet Explorer** _ის შემადგენლობაში შემავალი **Common Controls** ობიექტების გამოყენების საშუალება.
- კომპილატორის დონეზე **COM** ტექნოლოგიის გამოყენების საშუალება.
- **OLE DB** შაბლონების შექმნის საშუალება.
- მონაცემთა ბაზებთან მუშაობის საშუალება.
- **Active Document** კონტეინერების შექმნის საშუალება.
- ინტერნეტ აპლიკაციისათვის საჭირო **CHTTP** და **CInternet MFC**-კლასები.
- **MFC** და **ATL** კლასების ბაზაზე **ActiveX** კომპონენტების შექმნის საშუალება.
- უკვე არსებული **ActiveX** კომპონენტების ფართო არჩევანი.
- **AutoCAD**–ში ინტეგრირებული მოდულების შექმნისათვის საჭირო **ObjectARX** კლასების გამოყენების საშუალება.

The Object Model In AutoCAD

AutoCAD_ის ობიექტების მოდელი აღწერს, იმ ყველა ობიექტების, უნიკალურ და მკაცრად განსაზღვრულ იერარქიას, რომლებსაც იყენებს AutoCAD_ი. Autocad_ის ობიექტები შეიძლება დავყოთ სამ ძირითად კატეგორიად :

- ობიექტები რომლებიც აკონტროლებენ AutoCAD_ის ინტერფეისს
- ობიექტები რომლებიც მომხმარებელს ზუსტი ხაზვის საშუალებას აძლევენ (მათ არ გააჩნიათ ვიზუალური რეალიზაცია)
- გრაფიკული ობიექტები რომლებსაც მომხმარებელი იყენებს ნახაზის ვიზუალური ნაწილის შესაქმნელად. ისეთი ობიექტები როგორიცაა Line, Arc, Text და სხვა [3]

AutoCAD_ის ობიექტების მოდელი, რომელიც წარმოდგენილია ქვემოთ სტრუქტურული სახით, მომხმარებელს საშუალებას აძლებს ნახოს ობიექტების განლაგება და მათთან მიღწევის გზები. პირველ კატეგორიაში შემავალი ობიექტები გამოხატულია ყვითელი ფერით. მეორე კატეგორიაში შემავალი ობიექტები შეფერილია მწვანედ, ხოლო მესამე კატეგორიაში შემავალი გრაფიკული და ხაზვის ობიექტები შეფერილია ცისფერი ფერით.



AutoCAD_ის ობიექტების მოდელის სტრუქტურული გამოსახულება

განვიხილოთ AutoCAD_ის ობიექტები უფრო დაწვრილებით:

ჯგუფები და ობიექტები

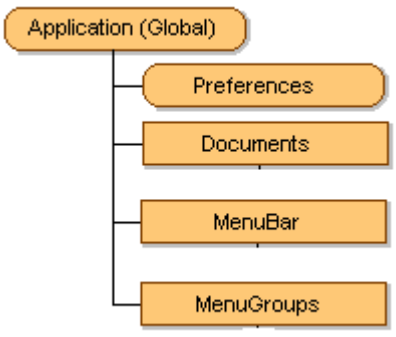
როგორც ზედა სქემიდან ჩანს, AutoCAD_ი და მის გარემოსი შექმნილი ნახაზები შედგება მრავალი სხვადასხვა ტიპის ობიექტებისაგან. ობიექტები დალაგებულია ჯგუფების მიხედვით. სქემაში ჯგუფები გამოსახულია მართკუთა ოთხკუთხედების სახით, ხოლო მასში შემავალი ობიექტები მომრგვალებული ფიგურების სახით.

Block, Group, SelectionSet, ModelSpase და PaperSpase ჯგუფები შეიცავენ სხვადასხვა ტიპის მრავალ ხაზვის ობიექტს, ისეთებს როგორიცაა Line, Arc, Text და სხვა. ყველა სხვა ჯგუფი შეიცავს მხოლოდ ერთი ტიპის ობიექტებს. მაგალითად ჯგუფი View შეიცავს მხოლოდ View ტიპის ობიექტებს.

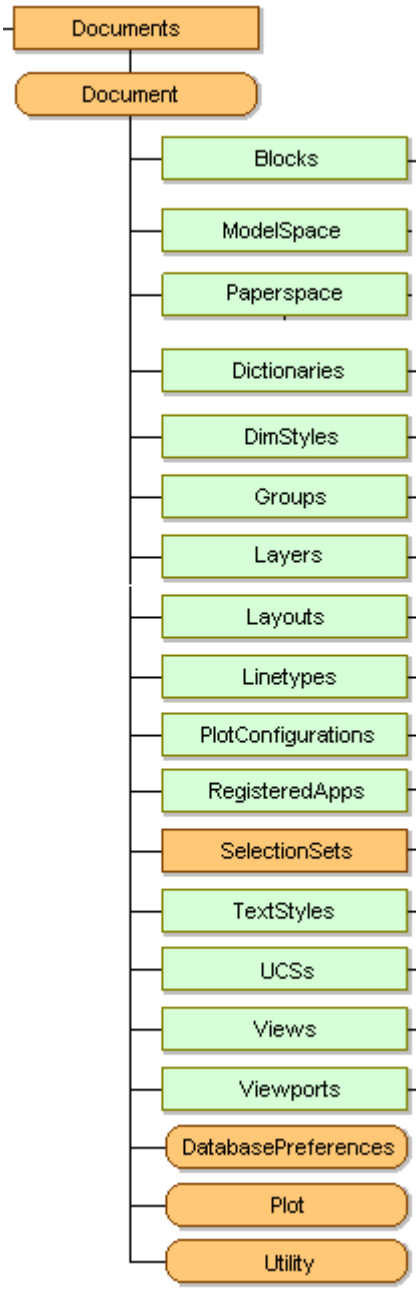
AutoCAD_ის ჯგუფები ეგრეთწოდებული ნულზე დამყარებული ჯგუფებია, რაც იმას ნიშნავს, რომ ჯგუფებში შემავალი ობიექტები დანომრილია ნულიდან ზემოთ. მაგალითად ჯგუფი Layer ყოველთვის შეიცავს ობიექტს (ფენას) სახელწოდებით "0".

ობიექტი Application

ეს ობიექტი AutoCAD_ის ძირითადი ობიექტია. მას გააჩნია თვისებები, მეთოდები და პროცედურები, რომლებიც აშკარად მეტყველებენ იმაზე რომ Application ობიექტი განასახიერებს მთლიან AutoCAD_ის აპლიკაციას.



ობიექტი Documents



ეს ობიექტი განასახიერებს AutoCAD_ის დოკუმენტს/ნახაზს. ერთი სამუშაო სესიის დროს შესაძლებელია რამოდენიმე ასეთი ნახაზის გახსნა და ისინი ერთად შეადგენენ ჯგუფს Documents. ვინაიდან Document ობიექტი წარმოადგენს AutoCAD_ის ნახაზს მისი საშუალებით შესაძლებელია გრაფიკული ობიექტებით და ხაზვის სტილებით მანიპულირება.

ხაზვის არეები (The Drawing Spaces)

Auto_ს გააჩნია ხაზვის ორი არე: Model Space და Paper Space.

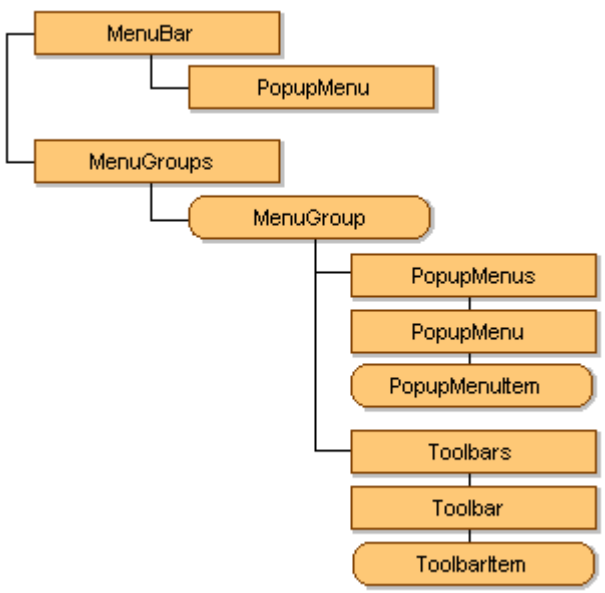
Model Space წარმოადგენს ხაზვის იმ არეს რომელშიც მომხმარებელი ქმნის ნახაზს და იგი შეიცავს ყველა ხაზვის ობიექტს, რომლებიც გამოიყენება ნახაზის შესაქმნელად.

Paper Space არეს მომხმარებელი იყენებს ნახაზის ამოსაბეჭდად გამზადების და ნახაზე შესაბამი დოკუმენტაციის დატანის მიზნით.

ბლოკები (Blocks)

ბლოკები AutoCAD_ში გამოიყენება სხვადასხვა ობიექტების ერთ ობიექტში გაერთიანების მიზნით. რამოდენიმე ობიექტისგან შემდგარი ასეთი ბლოკი ანუ უკვე ახალი ობიექტი შეიძლება იოლად ჩაისვას ნახაზში ან ახალ ბლოკში. ბლოკების გამოყენება ხაზვის ერთერთი რაციონალური მეთოდია, რომელიც ფართოდ გამოიყენება. შექმნილ ბლოკ გააჩნია თავისი თვისებები და ამ თვისებების ცვლილება ვრცელდება ყველა ერთი ტიპის ბლოკზე.

მენიუები და თულბარები (Menus end Toolbars)



AutoCAD_ს გააჩნია საკუთარი მენიუების და თულბარემბის პროგრამულად მოდიფიცირების საშუალება. იმ ობიექტების, რომლეთა მეშვეობით ხორციელდება ეს პროცესი, იერარქიული სტრუქტურა მოყვანილია გვერდითა ნახაზში.

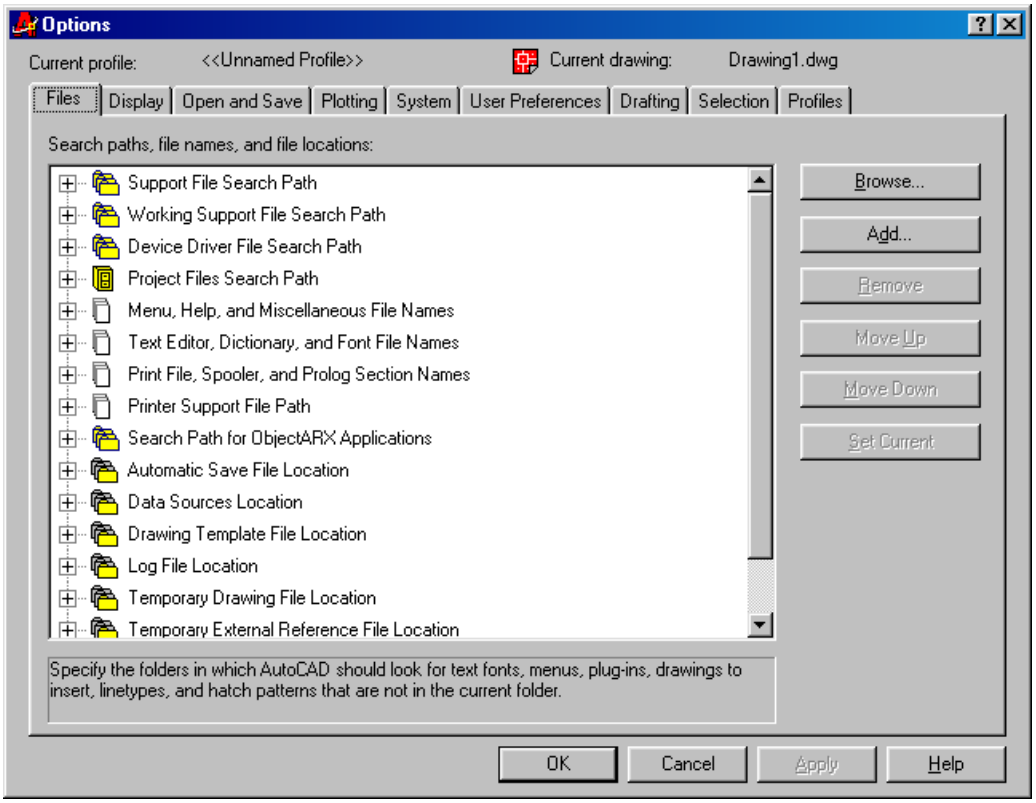
ჯგუფი MenuBar შეიცავს ყველა იმ მენიუს, რომელიც წარმოდგენილია PopupMenu ობიექტის სახით და ასახავს AutoCAD_ის ყველა მენიუს.

MenuGroup ობიექტი, რომელიც მოთავსებულია MenuGroup ჯგუფში, განასახიერებს იმ მენიუებისა და თულბარების ერთობლიობას რომლებიც ჩატვირთულია AutoCAD_ის სესიაში. მომხმარებელს შესაძლებლობა აქვს იმდენი მენიუ ჩატვირთოს რამდენიც მას სურს. თითოეული მენიუ და თულბარი წარმოდგენილია PopupMenu და Toolbar ობიექტების მეშვეობით და ამ

ობიექტების გავლით მომხმარებელს ხელი მიუწვდება ნებისმიერი თულბარის ან მენიუს ცალკეულ ღილაკზე.

ობიექტი Preferences

ეს ობიექტი გამოიყენება იმის კონტროლისათვის თუ რა გზებით და საშუალებებით მუშაობს მომხმარებელი AutoCAD_is გარემოში. მაგალითად მომხმარებელს შეუძლია განსაზღვროს AutoCAD_ის ობიექტების, სამუშაო არის და სხვა ვიზუალური და არავიზუალური ობიექტების თვისებების სხვადასხვა ვარიანტები, ის თუ რომელი დირექტორიიდან ჩაიტვირთოს AutoCAD_ში დამმხმარე პროგრამები და სხვა. Preferences ჯგუფში შემავალი ობიექტების თვისებების დათვალიერება და რედაქტირება ხდება Options dialog დიალოგური ფანჯრის მეშვეობით.



Options dialog დიალოგური ფანჯრა

ობიექტი Dictionaries

ჯგუფი Dictionaries შეიცავს ყველა იმ Dictionary ტიპის ობიექტს რომელიც ხელმისაწვდომია AutoCAD_ში. ეს არ არის მართლწერის შემოწმების ობიექტი, ის გამოიყენება ნახაზში ტექსტური ინფორმაციის დატანვა/შენახვისათვის. მაგალითად Dictionaries ჯგუფში შემავალი ობიექტი XRecord გამოიყენება ისეთი ინფორმაციის შესანახად როგორიცაა ობიექტის ID მისამართი და სხვა.

ზომები (Dimention) და ტექსტური სტილები (Text Styles).

AutoCAD_ის მომხმარებლისთვის აუცილებელია ნახაზზე ტექსტური და ზომითი ინფორმაციის დატანა. ზომების გამოტანა ეკრანზე, მათი რედაქტირება და შენახვა ხდება Dimension ობიექტის მეშვეობით. ხოლო ტექსტური ინფორმაციის გამოსახვის სტილები იმართება TextStyle ობიეცტის მიერ.

ფენები (Layer) და ხაზის ტიპები (Line types).

AutoCAD_ის ნახაზი ჩვეულებრივ შეიცავს რამოდენიმე ფენას (Layer) რაც ნახაზის ლოგიკურად განაწილების, სხვადასხვა ინფორმაციის სხვადასხვა ფენაში განლაგების საშუალებას იძლევა. მაგალითად ერთი ფენა გამოიყენება ნახაზის ძირითადი ობიექტების შესანახად, მეორე დამხმარე ხაზები და დატანილი ზომების შესანახად და ასე შემდეგ. ფენების მდგომარეობის მანიპულაცია კომპლექსური ნახაზის ადვილად ორგანიზების საშუალებას იძლევა. Layer ობიექტი წარმოადგენს ობიექტს რომლის მეშვეობითაც იმართება AutoCAD_ის ნახაზის ფენები. Layer ობიექტი მიეკუთვნება Layers ჯგუფს.

SelectionSet და Group ობიექტები.

AutoCAD_ში მონიშნული ობიექტები წარმოადგენენ ნახაზის ელემენტების დროებით ჯგუფს და ამ ჯგუფით მანიპულირება შეიძლება როგორც ერთი ობიექტით. ეს ჯგუფი AutoCAD_ში წარმოდგენილია SelectionSet ობიექტის საშუალებით. მომხმარებელს საშუალება აქვს შექმნას SelectionSet ტიპის ობიექტების ნებისმიერი რაოდენობა და ისინი გაერთიანებული იქნებიან SelectionSet ჯგუფში. SelectionSet ტიპის ობიექტები არ ინახება AutoCAD–ის ნახაზის შენახვის (დასეივების) დროს, აქედან გამომდინარე ეს ობიექტები არასტაბილურია.

Group ობიექტი SelectionSet ობიექტის იდენტურია, მხოლოდ იმ განსხვავებით რომ Group ტიპის ობიექტების შენახვა შესაძლებელია AutoCAD_ის ნახაზთან ერთად. Group ტიპის ობიექტები მიეკუტვნება Groups ჯგუფს.

ფორმატი (Layout).

ფორმატი, რომელიც წარმოდგენილია Layout ობიექტის საშუალებით, AutoCAD_ის ფურცლის სივრცის ექვივალენტურია. ფორმატი (Layout) ძირითადად წარმოადგენს AutoCAD_ის ერთ სუფთა ფურცელს რომელსაც გააჩნია უნიკალური სახელი. AutoCAD_ი ავტომატურად ქმნის ერთ Model Space ფორმატს სახელად Model და ერთ Paper Space ფორმატს სახელად Layout1, თუმცა მომხმარებელს შეუძლია იმდენი ფორმატის (Layout_ის) შექმნა რამდენიც სურს.

Plot და PlotConfiguration ობიექტები.

Plot ობიექტი წარმოადგენს პლოტერზე ბეჭდვის (Ploting) მეთოდებისა და თვისებების ერთობლიობის მართვის ობიექტს. მაგალითად მომხმარებელს სასუალება აქვს წინასწარ, Plot Preview_ს საშუალებით, ნახოს და/ან მოდიფიცირება გაუკეთოს ამოსაბეჭდად გამზადებულ ნახაზს. როდესაც ხდება ნახაზის ამობეჭდვა მისი ვიზუალური სახე დამოკიდებულია პლოტერის კონფიგურაციაზე. PlotConfiguration ობიექტი ინახავს ინფორმაციას პლოტერის კონფიგურაციის შესახებ. მომხმარებელს შეუძლია სურვილის მიხედვით შექმნას PlotConfiguration ობიექტების ნებისმიერი რაოდენობა, თითოეულ მათგანს ექნება უნიკალური სახელი და ისინი გაერთიანებული იქნებიან ჯგუფ PlotConfigurations_ში.

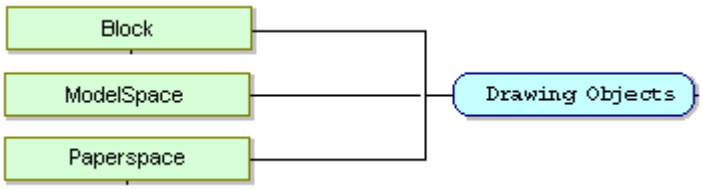
ობიექტი UCS.

ჯგუფი UCSs შეიცავს ყველა იმ UCS ობიექტს და/ან კოორდინატთა სისტემებს რომლებიც განსაზღვრული იყო მომხმარებლის მიერ AutoCAD–ში ნახაზის შექმნის დროს. UCS ობიექტების რაოდენობა არ არის შეზღუდული. UCS–ი წარმოადგენს მსოფლიო კოორდინატთა სისტემის (World coordinate System) კონკრეტული ამოცანისთვის უფრო მისაღებ კოორდინატთა სისტემად მიმოდიფიკაციას.

View და Viewports ობიექტები.

ჯგუფი Views შეიძლება შეიცავდეს View ობიექტების ნებისმიერ რაოდენობას და თითოეული ეს ობიექტი წარმოადგენს სამუშაო გარემოს სივრცის სხვადასხვა წარტილიდან ხედვის გრაფიკულ გამოსახულებას. AutoCAD_ი შეიცავს მეთოდებისა და თვისებების ერთობლიობას რომლებიც მომხმარებელს საშუალებს აძლევს შექმნას, მოდიფიცირება გაუკეთოს ან წაშალოს ხედვის ფორმები.

ობიექტი The Drawing.



AutoCAD_ი შეიცავს გრაფიკული ობიექტების გარკვეულ რაოდენობას რომლებიც ქმნიან ნახაზის ვიზუალურ ნაწილს. მათ ერთობლიობას ხაზვის ობიექტების ერთობლიობა ეწოდება. ეს ობიექტები შეიცავენ ისეთ გეომეტრიულ ობიექტებს როგორიცაა რკალი, წრეწირი, ელიფსი, სამგანზომილებიანი და ორგანზომილებიანი ზედაპირები, ტექსტური და ზომითი კომენტარები და სხვა. ეს ობიექტები შეიძლება დამატებულ იქნან Block, Modelspace ან PaperSpace ჯგუფებში. AutoCAD_ი შეიცავს ამ ობიექტების რედაქტირების, დამატების და წაშლის მეთოდებსა და თვისებებს.

ობიექტი Registered Application.

ობიექტი Registered Application, რომელიც მიეკუთვნება ჯგუფს Registered Applications გამოიყენება AutoCAD_ში გარე, მომხმარებლის მიერ შექმნილი, აპლიკაციების ჩასატვირთად. მომხმარებლის აპლიკაცია (User Aplication) AutoCAD_ში რეგისტრირდება მისი სახელის AutoCAD_ში როგორც Registered Application ობიექტის დამატებით.

User Aplication

User Aplication (მომხმარებლის აპლიკაცია) წარმოადგენს მომხმარებლის მიერ შექმნილ პროგრამულ უზრუნველყოფას, AutoCAD_ის მოდულს, ან მაკრო პროგრამას, რომელსაც ქმნის მომხმარებელი windows, IDE_ს, AutoCAD_ის და Kernal პროცესორის ობიექტების, თვისებების, პროცედურების და კლასების გამოყენებით კონკრეტული სამომხმარებლო ამოცანის გადაწყვეტის მიზნით. როგორც ზემოთ ავღნიშნეთ, ადს–ის არქიტექტურის თვალსაზრისით, AutoCAD_ი წარმოადგენს "მშობელ" სისტემას, ხოლო შესაბამისად User Aplication წარმოადგენს "შვილ" სისტემას. User Aplication–ის შესაქმნელად AutoCAD_ი შეიცავს ორ დაპროგრამების გარემოს Visual Basic (VB) და Visual LISP (VLISP). ასევე არსებობს ARX კლასების ერთობლიობა რომლებიც გაერთიანებულია ObjectARX ტექნოლოგიის მეშვეობით და ისინი გამოიყენება AutoCAD–ის გარემოში ინტეგრირებული, მომხმარებლის აპლიკაციების (მოდულების) შესაქმნელად, VisualC++ ინტეგრირებული დაპროექტების სისტემის გამოყენებით. დაპროგრამების ეს გარემოებები გამოიყენებენ სხვადასხვა პროგრამირების ენებს:

(VB->Visual Basic, VLISP->Visual LISP, ObjectARX->VisualC++).

თითოეულ დაპროგრამების ენას გააჩნია როგორც უპირატესობები ასევე აურყოფითი მხარეებიც, აქედან გამომდინარე დაპროგრამების გარემოს შერჩევისას დამპროექტებელმა უნდა გაითვალისცინოს ბევრი ფაქტორი. დაპროექტების მიზნის შესაბანისად რეკომენდირებულია უფრო იოლად რეალიზებადი გადაწყვეტილებების მიღება. გამოცდილება გვიჩვენებს, რომ პროგრამირების ენის სიძლიერეზე ან მის უარყოფით მხარეებზე საუბარი შესაძლებელია შემდეგი

კრიტერიუმების გათვალისწინებით: დასმული ამოცანის სწრაფი რეალიზება, მათემატიკური ოპერაციების სწრაფი გადაჭრა, დაპროექტების სისწრაფე, დიალოგური ფანჯრების გამოყენებით ინტერფეისის შექმნის სასუალება, გრაფიკულ მონაცემთა ბაზასთან მუშაობის საშუალება.

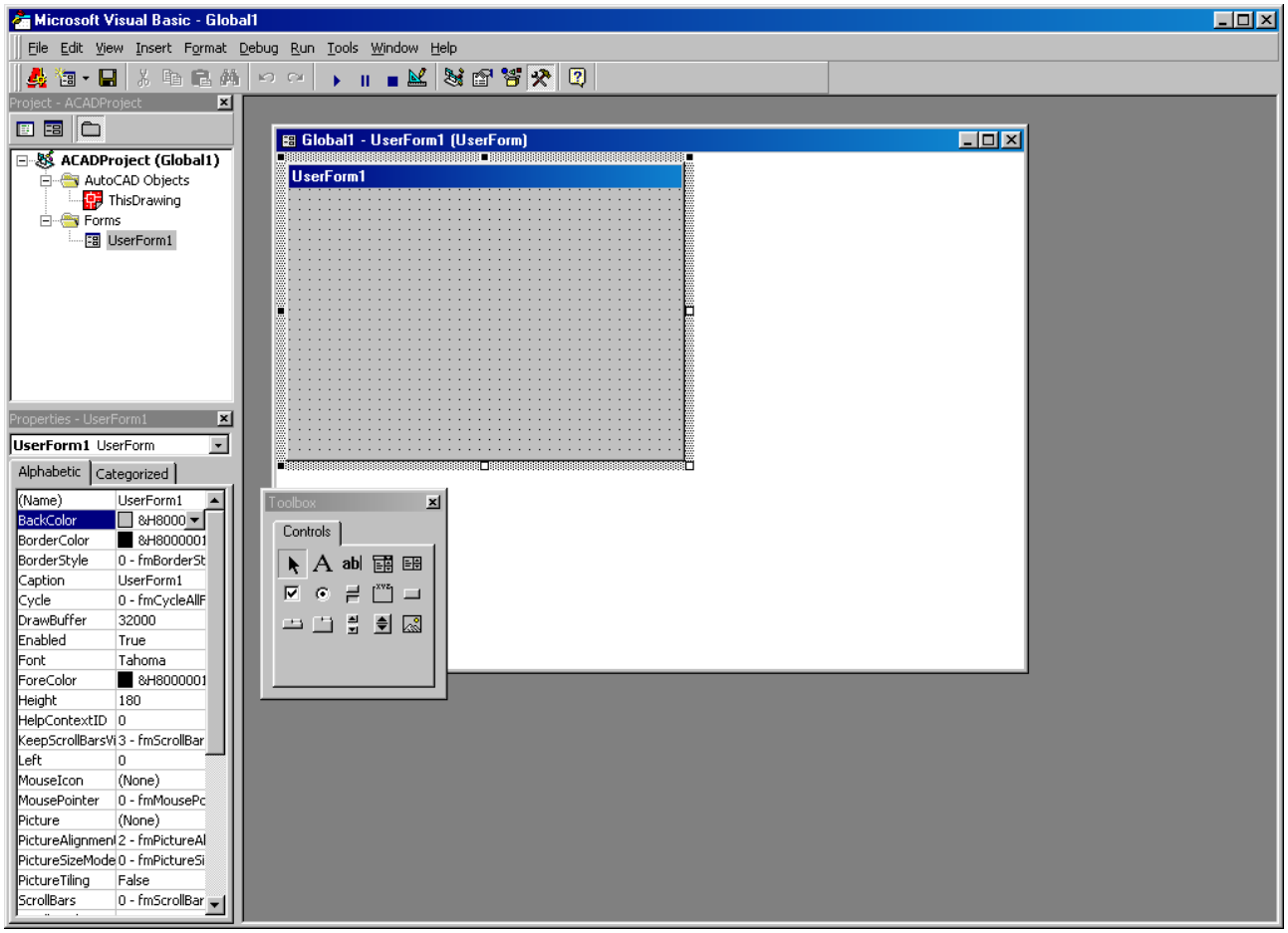
განვიხილოთ ზემოთ ჩამოთვლილი დაპროგრამების საშუალებები უფრო დაწვრილებით, მათი ერთმანეთთან შედარების მიზნით:

VB_ს მიმოხილვა

VB წარმოადგენს ვიზუალური დაპროგრამების ენას, რომელიც დაფუძნებულია სტანდარტულ **Visual Basic-**ზე. მომხმარებლის მიერ საკუთარი პროგრამული უზრუნველყოფის შესაქმნელად იგი იყენებს ფირმა **Microsoft** – ის მიერ შემოთავაზებულ **ActiveX Automation** ტექნოლოგიას. **ActiveX** ტექნოლოგია მნიშვნელოვნად აჩქარებს და ამარტივებს პროგრამული უზრუნველყოფის შექმნას, რადგან შესაძლებელი ხდება მზა ობიექტების გამოყენება მათი ფორმაზე თუ ნახაზზე გადატანის გზით.

Visual Basic_ის ძირითადი თვისებები:

- **VB**-ს მოდულები ინახება ე. წ. პროექტში. AutoCAD–ი საშუალებას იძლევა შეინახოს **VB**-ს პროექტი როგორც დოკუმენტის ნაწილი (**dwg** ფაილში) ან როგორც დამოუკიდებელი ფაილი (**dvb** გაფართოებით), რომელსაც გლობალური პროექტი ეწოდება. ამ ორი ტიპის პროექტთან ურთიერთობის დასამაყარებლად **AutoCAD**-ს გააჩნია **VB**-ს მენეჯერი, რომლის გამოძახება შესაძლებელია მენიუდან - **Tools | Macro | VBA Manager.**



Visual Basic-ს მენეჯერი

ეს ინსტრუმენტი იძლევა შემდეგი ოპერაციების ჩატარების საშუალებას:

- **dwg** ფაილში არსებული პროექტის ექსპორტირება გლობალური პროექტის ფორმატში და პირიქით;
- ორივე ტიპის ახალი პროექტის შექმნა;
- გლობალური პროექტის ჩატვირთვა და ამოტვირთვა ოპერატიული მეხსიერებიდან;
- ჩატვირთულ პროექტში არსებული მაკროსის ნახვა და გაშვება;

- **Visual Basic-**ის რედაქტორის გაშვება.

უტილიტას განთავსება, რომელიც დამუშავებულია სხვადასხვა ნახაზებთან სამუშაოდ, რა თქმა უნდა უმჯობესია გლობალურ პროექტში. აღსანიშნავია რომ **AutoCAD_**ი გაშვებისას არ ახდენს ავტომატურად **VB**-ს ინიციალიზაციას. თუმცა არსებობს მეთოდი, რომლის გამოყენებითაც **VB**-ს კოდი ავტომატურად ჩაიტვირთება ოპერატიულ მეხსიერებაში. ამისათვის უნდა შეიქმნას **VB**-ს გლობალური პროექტი სახელწოდებით **acad.dvb,** რომელიც უნდა ჩაიწეროს **AutoCAD**-ის კატალოგში. ეს პროექტი ავტომატურად ჩაიტვირთება **AutoCAD**-ის გაშვებისას. თუ ამ პროექტში იქნება მაკროსი სახელწოდებით **AcadStartup,** იგი ავტომატურ-ად შესრულდება.

VB-ს ყველაზე მნიშვნელოვანი უპირატესობა დაპროგრამების სხვა ენებთან შედარებით არის ის, რომ იგი ფართოდ არის გავრცელებული მთელს მსოფლიოში. მომხმარებელს, რომელსაც შეუძლია **Visual Basic-**ის ან საოფისე პროგრამების **VB**-ს გარემოში მუშაობა, არ გაუჭირდება **AutoCAD**-ის **VB**-ში ორიენტირება მაშინ, როცა **AutoLISP**-ის ათვისებისათვის დასჭირდებოდა დაპროგრამების მისთვის სრულიად უცნობი ენისა და ტექნოლოგიის შესწავლა.

VB -ის უარყოფით თვისებას წარმოადგენს ის, რომ მას არ შეუძლია **AutoLISP** -თან სრულყოფილი ურთიერთობა. მაგალითად შეუძლებელია **VB**-დან **AutoLISP** -ში დაწერილი ქვეპროგრამის გამოძახება.

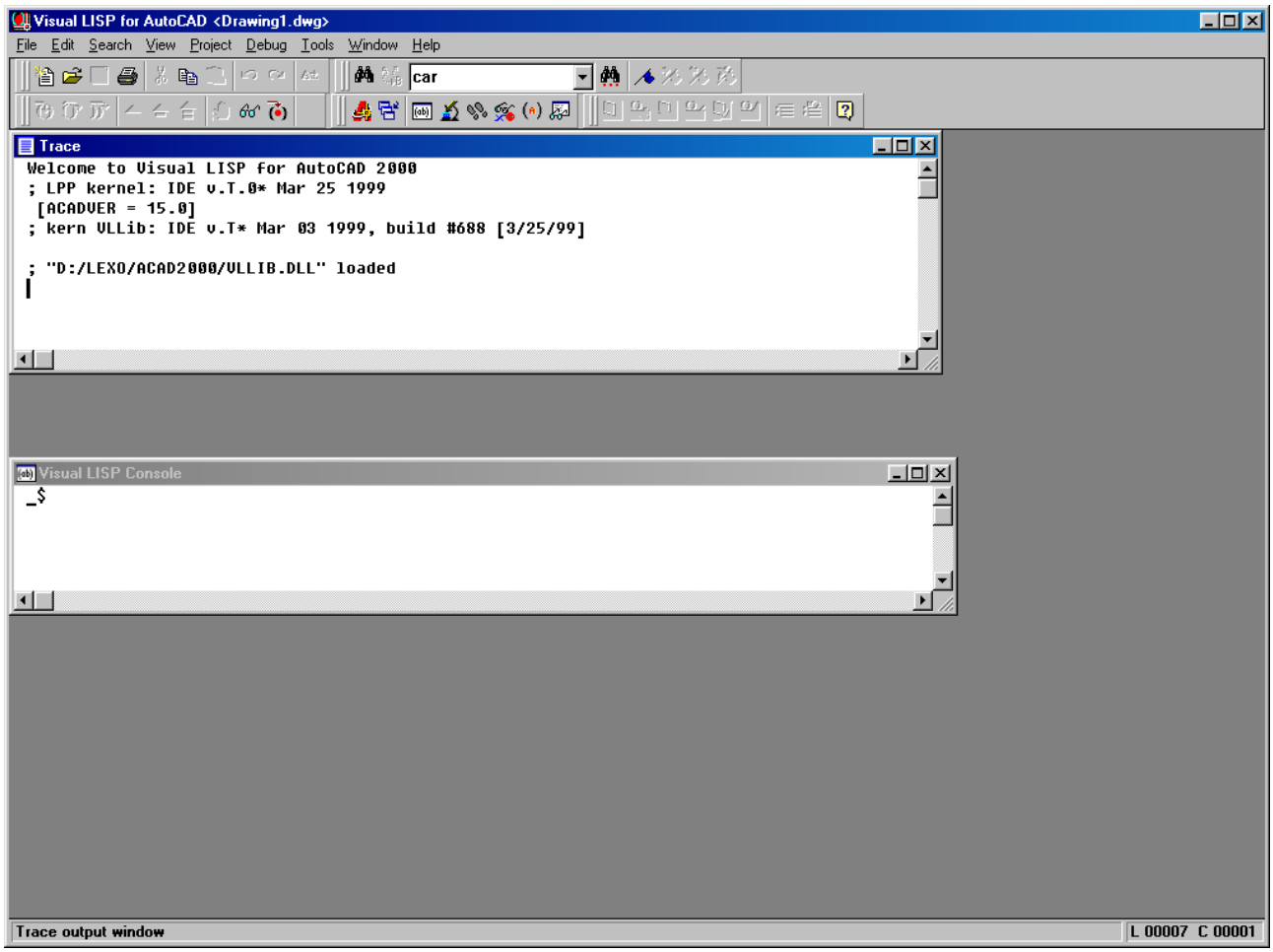
VB -ის ყველაზე მნიშვნელოვან უარყოფით თვისებად შეიძლება ჩაითვალოს ის რომ, **VB**–ს მეშვეობით შეუძლებელია იმ ობიექტებით მანიპულირება რომლებიც აკონტროლებენ AutoCAD_ის ინტერფეისს. VB_ში დაპროექტებისას შესაძლებელია AutoCAD_ის მხოლოდ ხაზვის ობიექტების, ფუნქციონალური ბრძანებების და VB_შივე შექმნილი დიალოგური ფანჯრების გამოყენება. უნდა აღინიშნოს ისიც,

რომ VB_ში შექმნილი დიალოგური ფანჯრები მოდალური ტიპისაა და მათი გააქტიურების შემდეგ მომხმარებელს არ შეუძლია AutoCAD_ის სხვა ფუნქციების გამოყენება. აქედან გამომდინარე, VB_ში შექმნილი პროექტის მომხმარებლის მოქმედების შესაძლებლობის არე შემოიფარგლება მხოლოდ პროექტში არსებული ბრძანებების და ფუნქციების გამოყენების შესაძლებლობით.

Visual LISP_ის მიმოხილვა

Visual LISP_ის მეშვეობით AutoCAD_ის მომხმარებელს შეუძლია AutoCAD_ის გარემოში საკუთარი აპლიკაციების შექმნა, რედაქტირება და ტესტირება. Visual LISP_ის ძირითადი თვისებები შეგვიძლია შემდეგნაირად ჩამოვაყალიბოთ:

- **მწარმოებლურობა.** Visual LISP_ი სასუალებას იძლევა შევქმნათ AutoCAD_ის გარემოში აპლიკაციები LISP დაპროგრამების ენის და ActiveX ობიექტების გამოყენებით. ActiveX ობიექტების გამოყენება საგრძნობლად ამცირებს დაპროექტების დროს და აქედან გამომდინარე ზრდის შრომისნაყოფიერებას.
- **გამოყენების სიმარტივე.** Visual LISP_ის დაპროგრამების გარემო შეიცავს:
  - → LISP_ის კოდის რედაქტორს ფუნქციათა სხვადასხვა ფერად გამოყოფის საშუალებით, რაც საგრძნობლად აადვილებს კოდის წაკითხვას.
  - → სინტაქსის შემოწმება სინტაქსური შეცდომების გამორიცხვის მიზნით.
  - → ავტოფორმატის გამოყენების საშუალება.
  - → სიმბოლური სახელების დინამიური დასრულება.



→ პროგრამის სტრუქტურაში თავისუფალი ნავიგაცია.

Visual LISP_ის დაპროგრამების გარემო

- **Windows_ში ინტეგრაცია.** Visual LISP_ი windows–ში ინტეგრაციის საშუალებას იძლევა შემდეგი ფუნქციებისა და დამხმარე საშუალებების მეშვეობით:
  - → ActiveX ობიექტების ინტერფეისი
  - → ოპერაციული სისტემის ფაილებთან მუშაობის დამხმარე ფუნქციები.

→ Visual LISP რედაქტორი, რომელიც აღიქვავს AutoCAD_ის გარემოში მიმდინარე პროცესებს.

Visual LISP_ის უპირატესობად სხვა, ზემოთ ჩამოთვლილ, დაპროგრამების ენებთან შედარებით, შეიძლება ჩაითვალოს ის, რომ Lisp დაპროგრამების ენა წარმოადგენს ხელოვნური ინტელექტის ენას და მასში ლოგიკური ოპერაციების დაპროგრამება მარტივად ხორციელდება.

Visual LISP_ის უარყოფით მხარედ შეიძლება ჩავთვალოთ:

- Visual LISP_ის გარემოში დაპროგრამების პროცესი ძირითადად შედგება AutoCAD–ის ბრძანებათა ხაზის (Command Line) დაპროგრამებისაგან.
- დიალოგური ფანჯრების შექმნა შეუძლებელია ეგრეთწოდებული Drag&drop ტექნოლოგიით. დიალოგური ფანჯრები იქმნება მხოლოდ Lisp კოდის გამოყენებით, რაც საგრძნობლად ზრდის დაპროექტების დროს და შრომატევადობას.
- შეუძლებელია იმ ობიექტებით მანიპულირება რომლებიც აკონტროლებენ AutoCAD_ის ინტერფეისს.
- მათემატიკური ამოცანების გადაჭრის დაბალი სიჩქარე.

ObjectARX_ის მიმოხილვა

ObjectARX წარმოადგენს კლასების ერთობლიობას, რომლებიც საშუალებს იძლევიან AutoCAD_ის ობიექტების გამოყენებით, შევქმნათ მომხმარებლის აპლიკაცია (მოდული), , VisualC++ ინტეგრირებული დაპროექტების სისტემის გამოყენებით.

Visual C++ (ObjectARX) -ზე დაპროგრამებას ზემოთ განხილულ საშუალებებთან შედარებით აქვს რამოდენიმე უპირატესობა. ObjectARX–ის ბიბლიოთეკები შეიცავენ ინსტრუმენტალური საშუალებების უნიკალურ ნაკრებს, რომლებიც საშუალებას იძლევიან გამოვიყენოთ AutoCAD, როგორც ობიექტზე ორიენტირებული არქიტექტურის დედა სისტემა და ვუზრუნველყოთ **პირდაპირი კავშირი** AutoCAD-ის მონაცემთა ბაზის სტრუქტურებთან, გრაფიკულ სისტემასთან და შიდა ბრძანებათა სისტემასთან.

ObjectARX გარკვეულ უპირატესობებს აძლევს დამპროექტებლებს და საბოლოო მომხმარებლებს. რაც ითვალისწინებს:

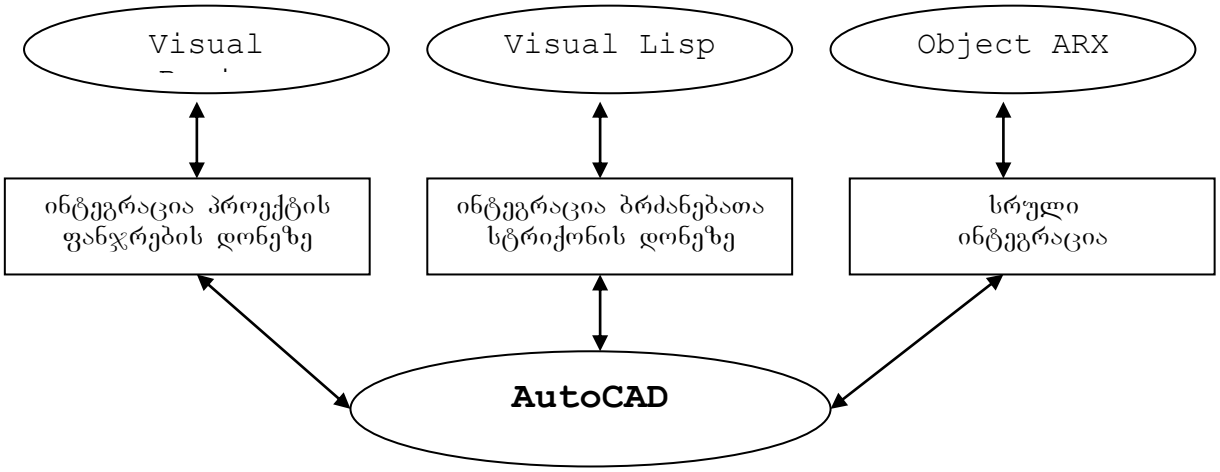
- AutoCAD_ის იმ ობიექტებით მანიპულირება რომლებიც აკონტროლებენ AutoCAD_ის ინტერფეისს.
- AutoCAD_ის პროგრამული მექანიზმების მართვის ფართო სასუალებები
- AutoCAD_ის მონაცემთა ბაზის ნებისმიერი ობიექტების გამოყენება და ახალი ობიექტების დამატების საშუალება მონაცემთა ბაზაში.
- ActiveX ტექნოლოგიით შექმნილი სპეციალიზირებული ობიექტების გამოყენების სასუალება

- ფოტორეალისტური რენდერი, კარკასული და მყარი ტანის სამგანზომილებიანი ობიექტების სწრაფი შექმნის სასუალება.
- პროგრამირების პროცესის გამარტივება ObjectARX Wizard უტილიტის გამოყენებით, რომელიც ავტომატურად ქმნის პროგრამული კოდის მნიშვნელოვან ნაწილს.

თვით კორპორაცია Autodesk_ი იყენებს ObjectARX ტექნოლოგიას საკუთარი Industry Foundation Classes (IFC) კლასების რეალიზაციისათვის და ისეთი ფართოდ ცნობილი პროგრამების შექმნისათვის როგორიცაა Autodesk Mechanical Desktop და AutoCAD Map. და რაც მთავარია ObjectARX ტექნოლოგია გამოიყენება Mechanical Application Initiative (MAI) პროგრამის ფარგლებში. ეს არის უნიკალური სტრატეგიული ალიანსი Autodesk–სა და პროგრამული უზრუნველყოფის შემქმნელებს შორის. MAI პარტნიორების პროგრამული პაკეტები საშუალებას გვაძლევს დვასრულოთ პროექტირების პროცესი – ანუ Mechanical Desktop_ის და AutoCAD_ის გარემოში შევასრულოთ სასრული ელემენტების ანალიზი, კინემატიკის და დინამიკის ანალიზი, NC პროგრამების შექმნა და მრავალი სხვა.

სხვა სისტემებისგან განსხვავებით, რომლებიც თავსებადობის საშუალებას იძლევიან მხოლოდ ფაილურ დონეზე, MAI პარტნიორების პროგრამები Mechanical Desktop–ის და AutoCAD–ის მომხმარებლის ინტერფეისს შეესაბამებიან, აქვთ პირდაპირი კავშირი Mechanical Desktop–ის და AutoCAD–ის ობიექტებთან, შეუძლიათ გამოიყენონ Mechanical Desktop–ის და AutoCAD_ის ატრიბუტები. ეს ღრმა ინტეგრაცია საშუალებას გვაძლევს გამჭოლი პროექტირების დასრულებულ გარემოს.

ზემოთ განხილული, AutoCAD_ის დაპროგრამების საშუალებების (VB, VLisp და ObjectARX), აღწერიდან და მათი ერთმანეთთან შედარებით შეიძლება ითქვას, რომ ამ დაპროგრამების საშუალებების ინტეგრაცია AutoCAD–ში გრაფიკულად შემდეგნაირად გამოიყურება:



ნახ №5 დაპროგრამების საშუალებების ინტეგრაცია AutoCAD_ში

AutoCAD_ის დაპროგრამების საშუალების ერთმანეთთან შედარებით აშკარად ჩანს ObjectARX ტექნოლოგიის უპირატესობა. როგორც ზემოთ ავღნიშნეთ, AutoCAD–ი უნივერსალური გრაფიკული სისტემაა, რომელიც განკუთვნილია ნებისმიერი სპეციალისტისთვის რომელიც მუშაობს ტექნიკურ თუ საპრეზენტაციო გრაფიკასთან. AutoCAD–ი წარმოადგენს მსოფლიოში ფართოდ გავრცელებულ საპროექტო-საკონსტრუქტორო სისტემა, რომელსაც იყენებენ მილიონობით დამპროექტებლები არქიტექტურაში, მანქანათმშენებლობაში, ელექტრომექანიკაში და სხვა მრავალ დარგში.

მეორე მხრივ სამანქანათმშენებლო ტექნოლოგიური პროცესების ავტომატიზებული დაპროექტება, ტრადიციულად გამოირჩევა გრაფიკული და კერძოდ კი გეომეტრიული ამოცანების მრავალფეროვნებითა და დიდი მოცულობით. ამიტომ, ამ ამოცანების გადაწყვეტა AutoCAD_ის რესურსებით, მომხმარებლის

აპლიკაციის დამუშავების გზით, წარმოადგენს აქტუალურ ამოცანას. თავის მხრივ, როგორც გამომდინარეობს ზემოთ ჩატარებული ანალიზიდან, AutoCAD_ის დაპროგრამების ObjectARX ტექნოლოგიას გააჩნია ყველაზე ფართო შესაძლებლობები AutoCAD_ის შიდა, Windows_ის და Kernal ბიბლიოთეკების ობიექტებთან მანიპულირებისა.

აქედან გამომდინარე წინამდებარე დისერტაციის მიზანია სამანქანათმშენებლო ტექნოლოგიური პროცესების ავტომატიზებული დაპროექტების ამოცანის გადაწყვეტის შესაძლებლობის გამოკვლევა AutoCAD_ის გარემოში ObjectARX ტექნოლოგიის ბაზაზე.

აღნიშნული ამოცანის გადასაწყვეტად შემუშავდა შემდეგი მეთოდური გეგმა:

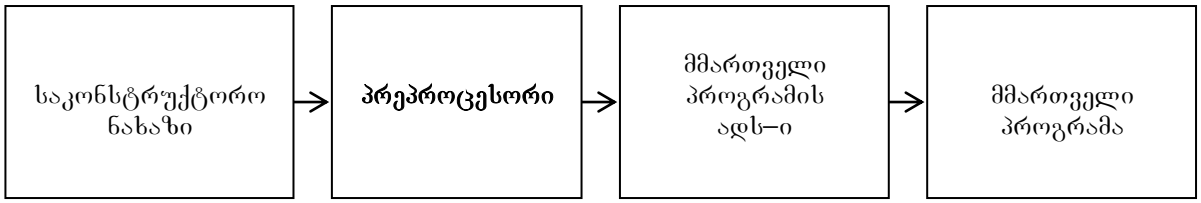
1. სამანქანათმშენებლო ტექნოლოგიური პროცესების ავტომატიზებული დაპროექტების ამოცანის ფორმულირება – დეტალის საწყისი აღწერის პრეპროცესორის სისტემის მიმოხილვა.
2. აღნიშნულ ამოცანასთან დაკავშირებული ObjectARX დაპროგრამების მეთოდის რესურსების გამოკვლევა.
3. პრეპროცესორის სისტემის ObjectARX ობიექტებით დაპროგრამების მეთოდის დამუშავება.
4. სისტემის პროგრამულ-მათემატიკური უზრუნველყოფის დამუშავება.

სამანქანათმშენებლო ტექნოლოგიური პროცესების ავტომატიზებული დაპროექტების ამოცანის ფორმულირება – დეტალის საწყისი აღწერის პრეპროცესორის სისტემის მიმოხილვა.

როგორც წინამდებარე დისერტაციის პირველ თავში ავღნიშნეთ, საკონსტრუქტორო ტექნოლოგიური დაპროექტება, როგორც პროცესი, ხორციელდება რამოდენიმე ატაპად:

- წინა საპროექტო კვლევა.
- გეომეტრიული მოდელირება.
- ფუნქციონალური მოდელირება.
- საინჟინრო ანალიზი.
- ნახაზის შედგენა/დოკუმენტაციის მომზადება.
- ტექნოლოგიური პროცესის დაგეგმვა.
- მმართველი პროგრამის გენერაცია.
- მმართველი პროგრამის გამართვა.

წინამდებარე დისერტაციის საკვლევი ამოცანის გადასაჭრელად შერჩეულ იქნა სამანქანათმშენებლო ტექნოლოგიური პროცესების ავტომატიზებული დაპროექტების ამოცანია რომელიც თავისთავად ითვალისწინებს საკონსტრუქტორო ტექნოლოგიური დაპროექტების ეტაპების (ნახაზის შედგენა/დოკუმენტაციის მომზადება და მმართველი პროგრამის გენერაცია) დამაკავშირებელი პრეპროცესორის სისტემის შექმნას. პრეპროცესორის ინტეგრაცია საკონსტრუქტორო ტექნოლოგიური დაპროექტების სისტემაში გრაფიკულად შემდეგნაირად გამოიყურება:

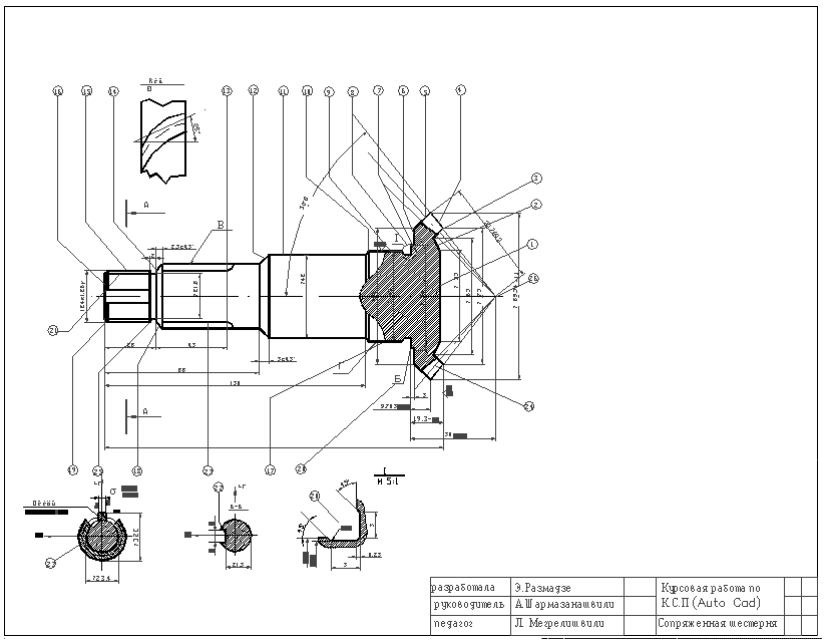


ნახ №6 მპ ადს-ის გაფართოებული არქიტექტურა

პრეპროცესორის შექმნა განპირობებულია შემდეგი ამოცანების გადაჭრის აუცილებლობით:

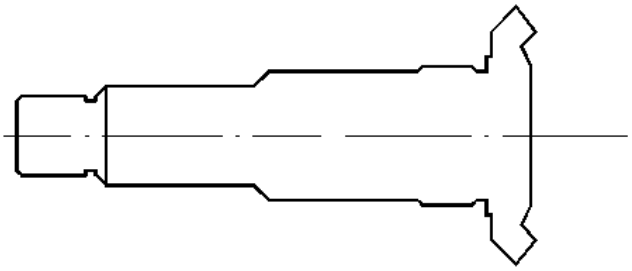
I. საკონსტრუქტორო ნახაზიდან ოპერაციული ნახაზის მიღების ამოცანა.

საკონსტრუქტორო ნახაზი წარმოადგენს დასამზადებელი ობიექტის კომპლექსურ ნახაზს რომელშიც გამოსახულია დასამზადებელი ობიექტი სხვადასხვა ხედში (ზედხედი, გვერდხედი და სხვა), ნახაზზე დატანილია დასამზადებელი ობიექტის გეომეტრიული ზომები (სიგრძე, რადიუსი, კუთხის ზომა და სხვა), დატანილია სიმეტრიის ღერძები, დაშტრიხვა, ტექსტური აღნიშვნები და სხვა, ანუ საკონსტრუქტორო ნახაზი მოიცავს სრულ ინფორმაციას (დოკუმენტაციას) დასამზადებელი ობიექტის შესახებ.



ნახ№7
საკონსტრუქტორო ნახაზი

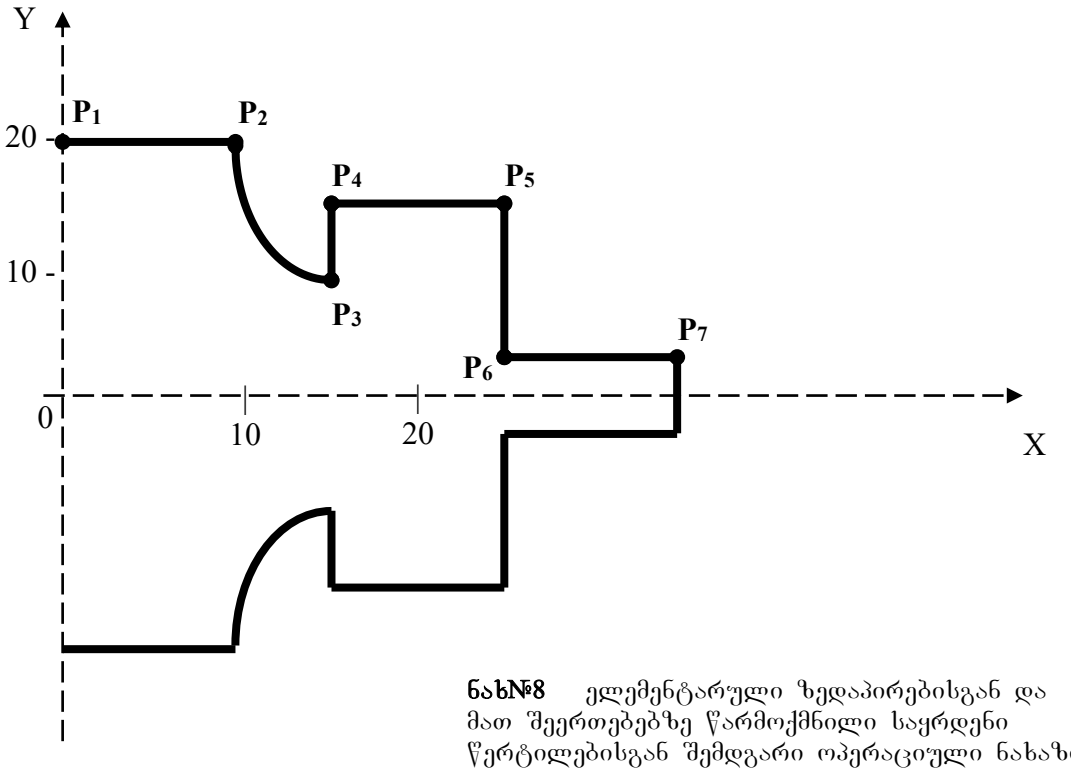
ოპერაციული ნახაზი კი აღწერს დეტალის მხოლოდ გეომეტრიულ ტოპოლოგიას.



ნახ№8 ოპერაციული ნახაზი

დეტალის გეომეტრიული ტოპოლოგია შეიძლება აღიწეროს რამოდენიმე გზით:

- ელემენტარული ზედაპირების შეერთებებზე წარმოქმნილი საყრდენი წერტილების თანმიმდევრობით და მათი კოორდინატებით ანუ "D ვექტორის" საშუალებით.



ნახ№8 ელემენტარული ზედაპირებისგან და მათ შეერთებებზე წარმოქმნილი საყრდენი წერტილებისგან შემდგარი ოპერაციული ნახაზი

თითოეულ საყრდენ წერტილს შეესაბამება სამი პარამეტრი:

1) X – საყრდენი წერტილის კოორდინატი OX ღერძის მიმართ.

2) Y – საყრდენი წერტილის კოორდინატი OY ღერძის მიმართ.

3) R – მოცემული საყრდენი წერტილიდან გამომავალი რკალის რადიუსის და მიმართულების პარამეტრი.

№8 ნახაზზე მოყვანილი კონტურის აღწერა ზემოთ აღნიშნული პარამეტრების მეშვეობით შემდეგნაირად გამოიყურება:

საყრდენი წერტილის №	X	Y	R
1	0	20	0
2	10	20	-20
3	15	10	0
4	15	15	0
5	25	15	0
6	25	5	0
7	35	5	0

როგორც ზემოთ ავღნიშნეთ ელემენტარული ზედაპირების შეერთებებზე წარმოწმნილი საყრდენი წერტილების კოორდინატები და მათი თანმიმდევრობა ქმნიან D ვექტორს.

ზემოთ მოყვანილი (ნახ№8) ოპერაციული ნახაზისათვის D ვექტორი შემდეგნაირად გამოიყურება:

$$\bar{D} = \begin{vmatrix} P1(X1,Y1,R1) \\ P2(X2,Y2,R1) \\ P3(X3,Y3,R1) \\ P4(X4,Y4,R1) \\ P5(X5,Y5,R1) \\ P6(X6,Y6,R1) \\ P7(X7,Y7,R1) \end{vmatrix}$$

სადაც :

P1,P2,...,Pn საყრდენი წერტილებია

X1,X2,...,Xn საყრდენი წერტილების კოორდინატები OX ღერძის მიმართ.

Y1,Y2,...,Yn საყრდენი წერტილების კოორდინატები OY ღერძის მიმართ.

R1,R2,...,Rn რადიუსები

- სპეციალური (ტექნოლოგიური პროგრამირების) ენების გამოყენებით. ამ ენების ტიპიურ მაგალითს წარმოადგენს APT (Automated Programing Tool) ენა. APT-ის ენაზე პროგრამის შექმნა ხელმისაწვდომია სპეციალისტების ფართო წრისათვის, საკმარისია მხოლოდ მათემატიკური სიმბოლიკის, გეომეტრიული აღწერის ხერხებისა და დაპროგრამების საფუძვლების ცოდნა.

APT-ის ენაში არსებობს ოპერატორების ოთხი ტიპი:

1. გეომეტრიული ოპერატორები. მათი საშუალებით აღიწერება ის გეომეტრიული ელემენტები, რომელთაგანაც შედგება დეტალი.
2. გადაადგილების ოპერატორები. გამოიყენება მუშა ორგანოს გადადგილების ტრაექტორიის აღწერისათვის.
3. პოსტპროცესორის ოპერატორები. გამოიყენება კონკრეტული ჩარხისა და მისი მართვის სისტემის შესაბამისი მახასიათებელი პარამეტრების გადასაცემად.
4. დამხმარე ოპერატორები. ეს არის სხვადასხვა ტიპის ბრძანებები, რომლებსაც იყენებენ დეტალისა და ინსტრუმენტების ტიპების, აგრეთვე სხვადასხვა დაშვებების განსაზღვრისათვის.

APT-ის ენის გეომეტრიულ ოპერატორს აქვს ასეთი სახე:

სიმბოლო = გეომეტრიული ტიპი / აღწერითი ხასიათის ინფორმაცია

ასეთი ოპერატორის მაგალითია:

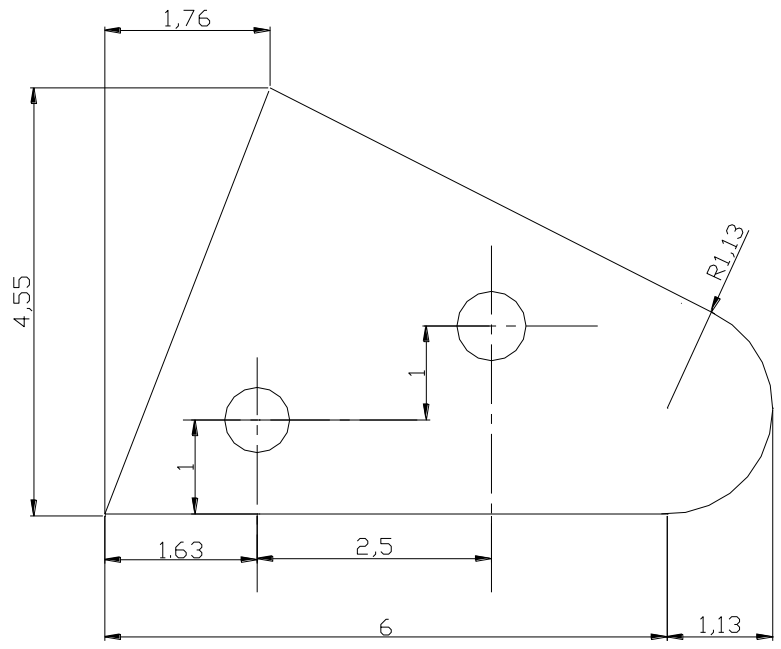
P1=POINT/5.0,4.0,0.0

ოპერატორი შედგება სამი ნაწილისაგან. პირველი წარმოადგენს გეომეტრიული ელემენტის აღმნიშვნელ სიმბოლოს. გეომეტრიულ ოპერატორის მეორე ნაწილად გვევლინება APT-ის ენის ბრძანება, რომელიც განსაზღვრავს გეომეტრიული ელემენტის ტიპს. გარდა ბრძანებისა POINT (წერტილი), APT-ის ენის ლექსიკონში შედის სხვა გეომეტრიული ელემენტების აღწერის ბრძანებები. მაგალითად: LINE (მონაკვეთი), PLANE (სიბრტყე), CIRCLE (წრეწირი). გეომეტრიული ოპერატორის მესამე ნაწილი შეიცავს ინფორმაციას, რომელიც ზუსტად, სრულად და ცალმხრივად აღწერს კონკრეტულ ელემენტს.

ნახ.№9–ზე მოყვანილია დეტალის ნახაზი, რომლის კონტური შედგება ელემენტარული ზედაპირებისაგან (ხაზებისა და რკალებისაგან). მაგალითისათვის ქვემოთ მოყვანილია ამ დეტალის აღწერა APT ენის გეომეტრიული ოპერატორების დახმარებით.

APT-ის ენაზე შექმნილ პროგრამის ლისტინგს, რომელიც აღწერს დეტალის გეომეტრიულ პარამეტრებს, ასეთი სახე ექნება:

```
P1 = POINT/6.0, 1.13, 0

P2 = POINT/0, 0, 0

P3 = POINT/6.0, 0, 0

P4 = POINT/1.76, 4.5, 0

L1 = LINE/P2, P3

C1 = CIRCLE/CENTER, P1, RADIUS, 1.13

L2 = LINE/P4, LEFT, TANTO, C1

L3 = LINE/P2, P4

PL1 = PLANE/P2, P3, P4
```



ნახ. 9

როგორც ზემოთ ავღნიშნეთ ოპერაციული ნახაზი აღწერს დეტალის მხოლოდ გეომეტრიულ ტოპოლოგიას, სწორედ ეს აღწერილობა წარმოადგენს მმართველი პროგრამის ადს–ისათვის საწყის (შემავალ) ინფორმაციას. აქედან გამომდინარე საკონსტრუქტორო ნახაზიდან ოპერაციული ნახაზის მიღება წარმოადგენს მნიშვნელოვან ამოცანას. პრეპროცესორის დანიშნულებაა ზემოთ აღნიშნული ამოცანის გადაწყვეტა. საკონსტრუქტორო ნახაზიდან ოპერაციული ნახაზის მიღება ხორციელდება საკონსტრუქტორო ნახაზიდან დასამზადებელი ობიექტის მხოლოდ გეომეტრიული ტოპოლოგიის გამოყოფის გზით. გამოყოფა შეიძლება განხორციელდეს რამოდენიმე გზით:

ა) მომხმარებელი თვითონ, ვიზუალური შერჩევის გზით, ახორციელებს გეომეტრიული პრიმიტივების გამოყოფას. ბ) გამოყოფა ტიპიური კონტურის მიხედვით, რომლის დროსაც იდენტიფიკაციის პროცესორის საშუალებით ხდება საკონსტრუქტორო ნახაზიდან დეტალის ტიპიური კონტურების იდენტიფიკაცია და გამოყოფა. ტიპიური კონტურების ტოპოლოგიური აღწერილობა ინახება იდენტიფიკაციის პროცესორის ცოდნის ბაზაში და ეს აღწერილობა წარმოდგენილია ელემენტარული ზედაპირების შეერთებებზე წარმოქმნილი საყრდენი წერტილების თანმიმდევრობით და მათი კოორდინატებით ანუ “D ვექტორის” საშუალებით ან APT_ენის ოპერატორებით. გ) დეტალის გეომეტრიული ტოპოლოგიის გამოყოფა ტიპიური პრომიტივების გამოყოფის გზით. ამ დროს იდენტიფიკაციის პროცესორის საშუალებით ხდება საკონსტრუქტორო ნახაზიდან ტიპიური პრიმიტივების გამოყოფა და მათი ერთობლიობით დეტალის კონტურის შედგენა. ტიპიური პრიმიტივები ინახება იდენტიფიკაციის პროცესორის ცოდნის ბაზაში და ამ პრიმიტივებმა უნდა გამოხატონ დეტალის შემდეგი

ძირითადი თვისებები: 1)კონტურის სტრუქტურა, ანუ ის, თუ რა ტიპის ელემენტარული ზედაპირებისაგან შედგება დეტალის კონტური. 2) ელემენტარული ზედაპირების ურთიერთგანლაგება სივრცეში.[4]

II. ოპერაციული ნახაზის შემოწმება.

პრობლემა მდგომარეობს იმაში, რომ დეტალის ჭრის ზედაპირი, ოპერაციულ ნახაზში, უნდა შედგებოდეს ერთმანეთთან უწყვეტად დაკავშირებული პრიმიტივების ერთობლიობისაგან (წრფეების და/ან რკალების ერთობლიობისაგან). თუ ოპერაციულ ნახაზში ეს უწყვეტობა დარღვეული იქნება, ასეთი ნახაზის საფუძველზე შექმნილი (გენერირებული) მმართველი პროგრამა ტექნოლოგიური პროცესის არასწორ წარმართვას გამოიწვევს, რაც გამოიხატება ტექნოლოგიურ დანადგარზე (ჩარხზე) შეცდომის მიღებით. ტექნოლოგიურ დანადგარზე მიღებული შეცდომის აღმოფხვრა და ტექნოლოგიური პროცესის გამართვა კი საკმაოდ დიდ და შრომატევად უკუპროცესს წარმოადგენს, ვინაიდან შეცდომის აღმოჩენის შემთხვევაში გამოკვლეულ უნდა იქნას მისი გამომწვევი მიზეზები, საკონსტრუქტორო ნახაზში უნდა იქნას შეტანილი ცვლილებები, უნდა მოხდეს მმართველი პროგრამის ხელახალი გენერაცია და ტექნოლოგიური პროცესის გამართვა. ეს ყველაფერი კი დამატებითი, დროის და საწარმოო და საპროექტო, რესურსების დახარჯვას მოითხოვს. ნახაზში შეცდომა შეიძლება გამოწვეული იყოს რამოდენიმე მიზეზით: მხაზველის არაკომპეტენტურობა, ხაზვის პროცესში მექანიკური შეცდომა, სხვა და სხვა ნახაზის ერთ ნახაზში გაერთიანებით გამოწვეული შეცდომები და სხვა. აღნიშნული შეცდომებიდან გამომდინარე შეიძლება საერთოდ შეუძლებელიც კი გახდეს მმართველი პროგრამის მიღება. იმიტომ რომ საყრდენი წერტილების არათანხვედრა ანუ მათ შორის წარმოქმნილი სიცარიელე ვერ აღიქმება მმართველი პროგრამების ადს–ის მიერ. ზემოთ

თქმულიდან გამომდინარე, პრეპროცესორის მიერ ოპერაციულ ნახაზში პრიმიტივების უწყვეტობის შემოწმება და ამ უწყვეტობის აღდგენა ძალიან მნიშვნელოვან ამოცანას წარმოადგენს.

III. კოორდინატთა სისტემის ტრანსფორმაცია.

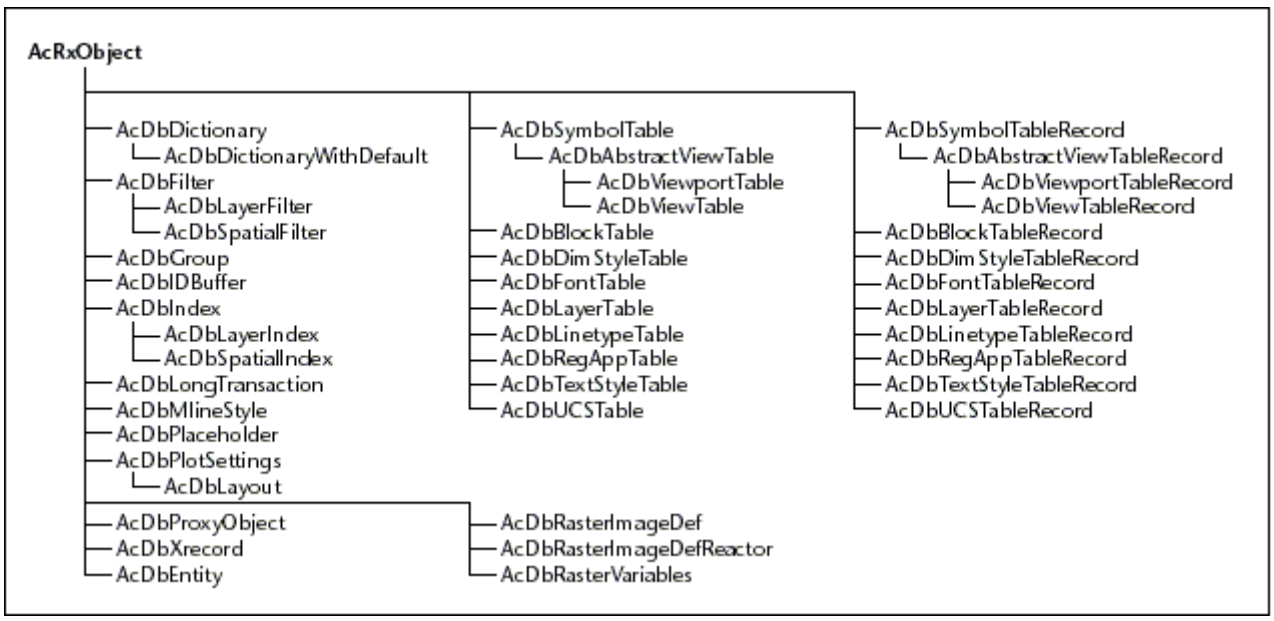
AutoCAD_ში ან სხვა პროექტირების (ხაზვის) პროგრამაში შექმნილი საკონსტრუქტორო ნახაზი, როგორც ზემოთ ავღნიშნეთ, შედგება პრიმიტივებისგან რომლებიც დახაზულია იმ კოორდინატთა სისტემაში, რომელიც გააჩნია ამათუიმ პროექტირების პროგრამას. მაგალითად AutoCAD_ში შეიძლება ნახაზის შექმნა WCS (world coordinate system), UCS(The user coordinate system) და სხვა კოორდინატთა სისტემაში. ყველა ტექნოლოგიურ დანადგარს კი გააჩნია თავისი კოორდინატთა სისტემა და მმართველი პროგრამის გენერაციის დროს დასამზადებელი დეტალის გეომეტრიული ზომები განიხილება ტექნოლოგიური დანადგარის კოორდინატთა სისტემაში. ამიტომ პრეპროცესორის დანიშნულებაა საკონსტრუქტორო ნახაზიდან მიღებული ოპერაციული ნახაზის კოორდინატთა სისტემის ტრანსფორმაცია ტექნოლოგიური დანადგარის (ჩარხის) კოორდინატთა სისტემაში. რაც თავისთავად ითვალისწინებს ოპერაციული ნახაზის კოორდინატთა სისტემის ნულოვანი წერტილის მიმართ განსაზღვრული პრიმიტივების კოორდინატების გაანგარიშებას, ტექნოლოგიური დანადგარის კოორდინატთა სისტემის ნულოვანი წერტილის მიმართ.

აღნიშნულ ამოცანასთან დაკავშირებული **ObjectARX** დაპროგრამების მეთოდის რესურსების გამოკვლევა.

წინა თავში აღნიშნული, პრეპროცესორის ამოცანების ObjectARX ტექნოლოგიით გადაწყვეტის მიზნით განვიხილოთ ObjectARX დაპროგრამების მეთოდის რესურსები უფრო დაწვრილებით. განვიხილოთ ObjectARX_ის კლასები რომელთა მეშვეობითაც შესაძლებელია AutoCAD_ის, Windows–ის და Kernal–ის ობიექტებით მანიპულირება.

- **AcDb** კლასების ბიბლიოთეკა წარმოადგენს AutoCAD_ის ობიექტების მონაცემთა ბაზასთან მუშაობისთვის განკუთვნილი კლასების ერთობლიობას. AutoCAD_ის ობიექტების მონაცემთა ბაზა შეიცავს სრულ ინფორმაციას AutoCAD_ის გრაფიკული (Line, Arc, Polyline და სხვა) და არაგრაფიკული (Layers, Linetypes, Text Styles და სხვა) ობიექტების შესახებ. დამპროექტებელს შეუძლია ამ ობიექტების გამოყენება, რედაქტირება და მონაცემთა ბაზაში ახალი ობიექტების დამატება.

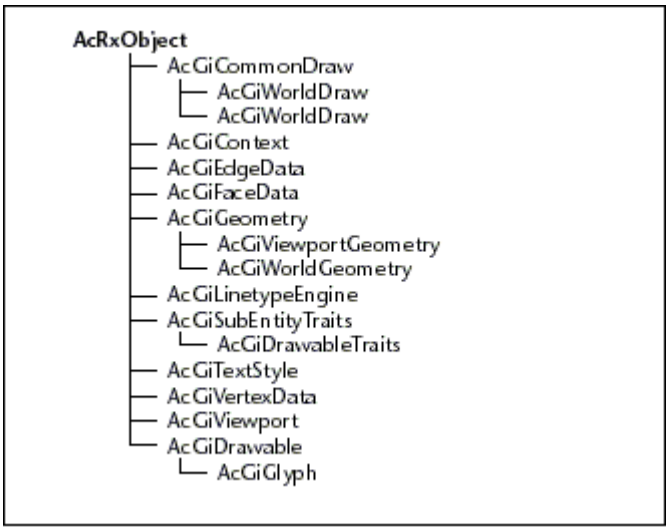
AcDb კლასების და ობიექტების იერარქია შემდეგნაირად გამოიყურება:



AcDb კლასების და ობიექტების გამოყენებით დამპროექტებელს შეუძლია AutoCAD_ის ისეთი ობიექტებით და პროცედურებით მანიპულირება, როგორიცაა: **AutoCAD_ის ობიექტების ბიბლიოთეკა** (AcDbDictionary და AcDbDictionarywithdefault კლასების გამოყენებით), **AutoCAD_ის ობიექტების ბიბლიოთეკაში შემავალი ცხრილები, ველები და ჩანაწერები** (AcDBSymbolTable და AcDBSymbolTableRecord კლასების და მათში შემავალი ქვეკლასების გამოყენებით), **ბლოკები** (AcDbBlockTable, AcDbBlockTableRecord), **ზომები** (AcDbDimStyleTable, AcDbDimStyleTableRecord), **ფონტი**(AcDbFontTable, AcDbFontTableRecord), **ტექსტის სტილები**(AcDbTextStyleTable, AcDbTextStyleTableRecord),**ფენები**(AcDbLayerTable,AcDbLayerTableRecord **ხაზის ტიპები**(AcDbLineTypeTable,AcDbLineTypeTableRecord), **UCS–ი**(AcDbUCSTable, AcDbUCSTableRecord), **ობიექტების და ბლოკების ფილტრები** (AcDbFilter, AcDbLayerFilter და AcDbSpatialFilter კლასების გამოყენებით), **ჯგუფები და ბუფერი** (AcDbGroup და AcDbBuffer კლასების გამოყენებით), **ობიექტების ინდექსები** (AcdBIndex და მასში შემავალი ქვეკლასების გამოყენებით), **ტრანზაქციები** (AcDbLongTransaction), **პლოტერის პარამეტრები** (AcDbPlaceholder, AcdbPlotSettings,AcDbLayout), **რასტრული გამოსახულებები** (AcDbRasterImageDef, AcDbRasterImageDefReactor, AcDbRasterVariables), **ტექსტური ჩანაწერები** (AcDbXrecord, AcDbProxyObject), **AutoCAD_ის ვიზუალური ობიექტები** (Line, arc, Polyline და სხვა, რომლებიც გაერთიანებულია AcDbEntity კლასში)

- **AcGi** კლასების ერთობლიობა წარმოადგენს AutoCAD_ში შექმნილი ობიექტების ვიზუალიზაციისთვის და რენდერისთვის საჭირო გრაფიკული კლასების ერთობლიობას. ამ კლასებს იყენებს AutoCAD_ის ისეთი ბრძანებები როგორიცაა: Render, viewport, shade და სხვა.

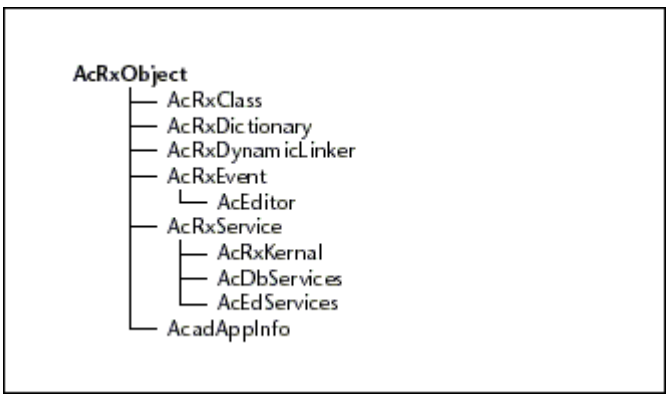
AcGi კლასების და ობიექტების იერარქია შემდეგნაირად გამოიყურება:



AcGi კლასების და ობიექტების გამოყენებით დამპროექტებელს შეუძლია: **გეომეტრიული ობიექტების ვიზუალიზაცია სხვადასხვა ხედვის ზონებისათვის** (AcGiCommonDrow და მასში შემავალი ქვეკლასების გამოყენებით), **ფარული ხაზების გამოჩენა/გაქრობა, რეგენერაცია** (AcGiContext, AcGiEdgeData, AcGiFaceData კლასების გამოყენებით), **გრაფიკული გამოსახულების ბუფერში შენახვა** (AcGiGeometry, AcGiWorldGeometry, AcGiViewportGeometry), **ინფორმაცია გეომეტრიული პრიმიტივების ფერების, ხაზის ტიპების, ჩრდილების ტიპების შესახებ** (AcGiLinetypeEngin, AcGiSubEntityTraits), **ტექსტის ატრიბუტები** (AcGiTextStyle), **ინფორმაცია კარკასული და ბადური ზედაპირების შესახებ** (AcGiVertexData), **ინფორმაცია viewport–ის შესახებ** (AcGiViewport), **რენდერი** (AcGiDrawable და ამ კლასში შემავალი ქვეკლასების გამოყენებით).

- **AcRx** კლასების ბიბლიოთეკა წარმოადგენს აპლიკაციის შექმნისთვის, იდენტიფიკაციისთვის და რეგისტრაციისთვის საჭირო კლასების და ობიექტების ერთობლიობას .AcRxObject კლასის ყველა ქვეკლასს გააჩნია ამ კლასების თვისებების აღმწერი ობიექტი, რომელიც შეიცავს ინფორმაციას კლასის და მასში შემავალი ობიექტების შესახებ და გამოიყენება კლასების და ობიექტების იდენთიფიკაციისათვის. აქედან გამომდინარე შესაძლებელი ხდება იმის შემოწმება თუ რომელ კლასს მიეკუთვნება ობიექტები, ხომ არ ემთხვევა ერთ კლასში შემავალი ობიექტების სახელები ერთმანეთს და სხვა.

AcRx კლასების და ობიექტების იერარქია შემდეგნაირად გამოიყურება:



AcRxClass კლასის საშუალებით დამპროექტებელს შეუძლია:

- C++_ის ობიექტების იდენტიფიკაცია და რეგისტრაცია.
- C++_ის არსებულ კლასებსი დამატებითი პროტოკოლების შექმნა
- ობიექტების თავსებადობის და ექვივალენტობის შემოწმება.
- C++_ის ობიექტების კოპირება

AcRxDictionary კლასის გამოყენებით შესაძლებელია AcRxObject კლასის ობიექტების მიმთითებლებით მანიპულირება(დამატება, წაშლა, რედაქტირება).

AcRxDynamicLinker კლასის გამოყენებით დამროექტებელი აკონტროლებს AutoCAD–ის ისეთ პროცესებს როგორიცაა ARX აპლიკაციის ჩატვირთვა/ამოტვირთვა.

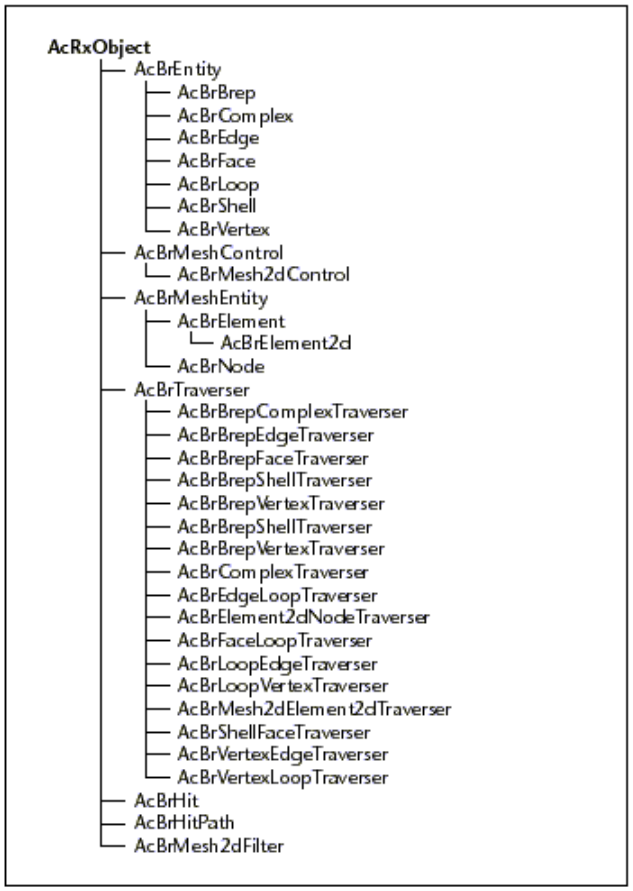
AcRxEvent და მასში შემავალი ქვეკლასების საშუალებით შესაძლებელია AutoCAD–ში მიმდინარე პროცესების კონტროლო.

AcRxService და ამ კლასში შემავალი ქვეკლასების გამოყენებით ხორციელდება ObjectARX მოდულებს შორის ინფორმაციის გაცვლა.

AcadAppInfo კლასი გამოიყენება AutoCAD_ში ჩატვირთული ObjectARX აპლიკაციების შესახებ ინფორმაციის შესანახად და დასამუშავებლად.

- **AcBr** კლასების ბიბლიოთეკა დამროექტებელს საშუალებას აძლევს მოიპოვოს ტოპოლოგიური, გეომეტრიული და ანალიტიკური ინფორმაცია AutoCAD–ში შექმნილი გეომეტრიული ობიექტების შესახებ.

  AcBr კლასების და ობიექტების იერარქია შემდეგნაირად გამოიყურება:



AcBr კლასი და მასში შემავალი ქვეკლასები დამპროექტებელს საშუალებას აძლევენ მოიპოვოს ტოპოლოგიური ინფორმაცია ობიექტების შესახებ.

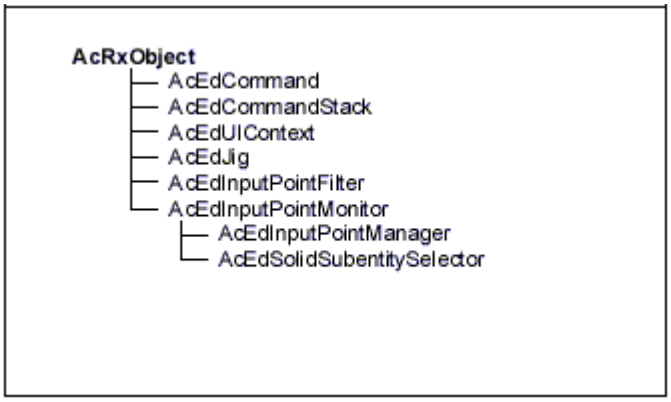
AcBrMeshControl, AcBrMeshEntity კლასები და ამ კლასებში შემავალი ქვეკლასები საშუალებას იძლევიან მოვიპოვოთ ინფორმაცია ბადური ზედაპირებისგან შექმნილი ობიექტების შესახებ.

AcBrTraverser კლასი და მასში გაერთიანებული ქვეკლასების დანიშნულებაა გეომეტრიული ობიექტების, კარკასული და ბადური ზედაპირებით შექმნილი ობიექტების გაერთიანებით მიღებული ტოპოლოგიების შესახებ ინფორმაციის მოპოვება და კონტროლი.

AcBrHit და AcBrHitpath კლასები იძლევიან ინფორმაციას ხაზობრივი ობიექტების (Line, Polyline, Mline და სხვა) საყრდენი წერტილების, გადაკვეთის წერტილების და სხვა ტოპოლოგიური მონაცემების შესახებ.

- **AcEd** კლასი და მასში შემავალი ობიექტები გამოიყენება AutoCAD–ში ბრძანებების რეგისტრაციისთვის და AutoCAD–ში მომხდარი მოვლენების აღქმისათვის. **AcEd** კლასი შეიცავს ქვეკლასებს და ობიექტებს რომელთა მეშვეობითაც დამპროექტებელს შეუძლია AutoCAD–ში შექმნას და დაარეგისტრიროს საკუთარი ბრძანებები, რომლებიც AutoCAD–ის ბრძანებების იდენტური იქნება, რაც იმას გულისხმობს რომ ამ ახლად შექმნილი ბრძანებების გამოყენება შესაძლებელი იქნება ნებისმიერი სხვა აპლიკაციის მეშვეობით. **AcEd** კლასი და მასში შემავალი ობიექტები ასევე იძლევიან AutoCAD–თან ინტერაქტივის საშუალებას.

AcEd კლასების და ობიექტების იერარქია შემდეგნაირად გამოიყურება:



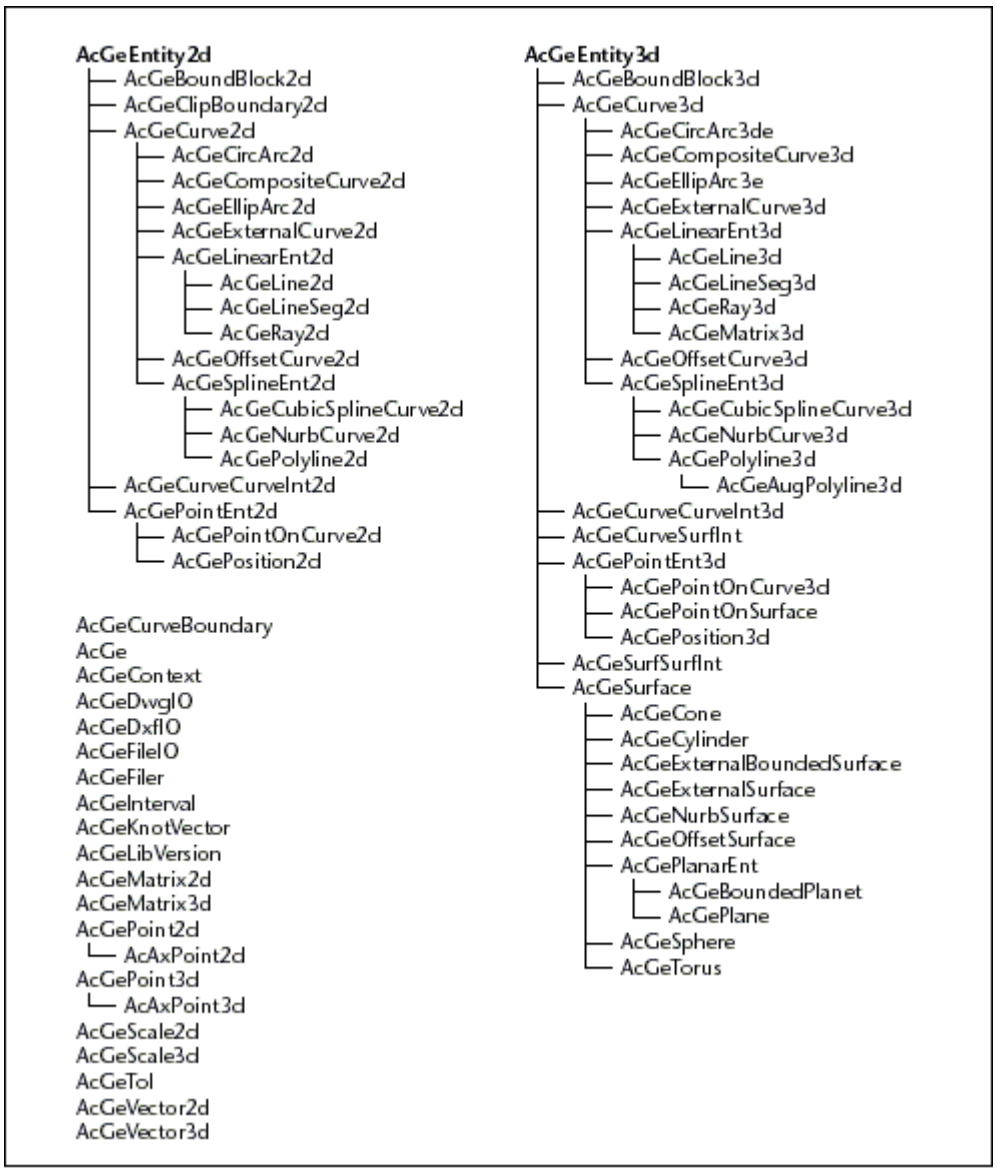
AcEd კლასების და მათში შემავალი ქვეკლასების გამოყენებით დამპროექტებელ შეუძლია: ObjectARX აპლიკაციიდან ბრძანება გადასცეს პირდაპირ AutoCAD–ის ბრძანებათა ხაზს (AcEdCommand, AcEdCommandStack კლასების გამოყენებით), AutoCAD–ის კონტექსტურ მენიუში საკუთარი პუნქტების დამატება (AcEdUIContext კლასის დახმარებით) , AutoCAD–ის შემავალი წერტილების (InputPointers)კონტროლი(AcEdInputPointFilter,AcEdInputPointmonitor და ამ კლასების ქვეკლასების მეშვეობით)

- **AcGe** კლასები გამოიყენება ალგებრული და გეომეტრიული ოპერაციების შესასრულებლად. **AcGe** კლასები წარმოადგენს **AcDb** კლასების დამხმარე კლასებს, რომლებიც გამოიყენება ორგანზომილებიანი და სამგანზომილებიანი გეომეტრიული ოპერაციების შესასრულებლად. **AcGe** კლასები შეიცავს ისეთ გეომეტრიულ ობიექტებს როგორიცაა: points, curves და surfaces.

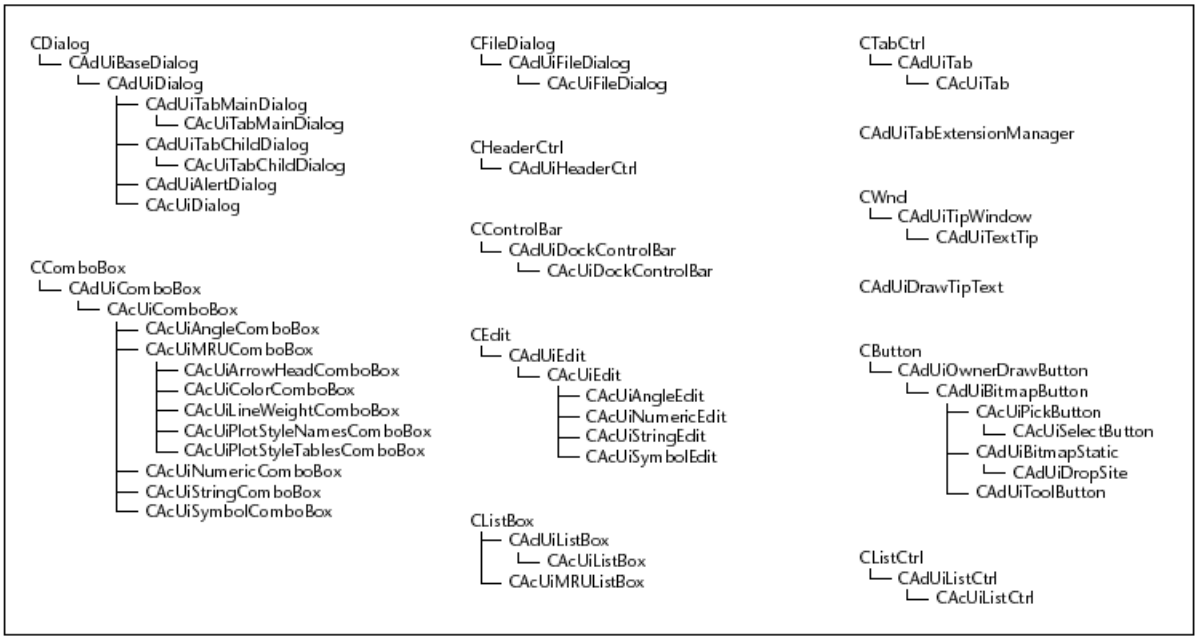
  AcGe კლასი შედგება ორი ძირითადი ტიპის ქვეკლასებისაგან: ორგანზომილებიანი გეომეტრიული ოპერაციების კლასებისაგან და სამგანზომილებიანი გეომეტრიული ოპერაციების კლასებისაგან. ძირითად კლსასებს წარმოადგენს AcGeEntity2d და AcGeEntity3d კლასები. თუმცა

არსებობს ისეთი კლასებიც რომლებიც არ წარმოადგენენ ამ კლასების ქვეკლასებს. მაგალითად ისეთი კლასები როგორიცაა AcGePoint2d, AcGeVector2d, და AcGeMatrix3d და სხვა. **AcGe** კლასების გამოყენებით შესაძლებელია, AutoCAD_ის გარემოში, პროგრამულად ის ოპერაციების ჩატარება როგორიცაა: მატრიცების მიმატება, გადამრავლება (AcGeMatrix3D, AcGeMatrix2D), კურვ ტიპის ობიექტების ურთიერთ განლაგების და ურთიერთ დამოკიდებულების განსაზღვრა (AcGeCurve2d, AcGeCurve3d, AcGeCCurveCurveInt2d, AcGeCCurveCurveInt3d, AcGeCircArc3d კლასების და ამ კლასებში შემავალი ქვეკლასების გამოყენებით), გეომეტრიული ობიექტების ზომების ცვლილება (AcGeScale2d, AcGeScale3d) და სხვა.

AcGe კლასების და ობიექტების იერარქია შემდეგნაირად გამოიყურება:



AcUI და **AdUI** კლასების ბიბლიოთეკები გამოიყენება ARX აპლიკაციების ინტერფეისის შესაქმნელად. ამ კლასების და მათში შემავალი ქვეკლასების გამოყენებით დამპროექტებელს შეუძლია შექმნას ინტერფეისი შემდეგი სტანდარტული ობიექტების გამოყენებით: დიალოგური ფანჯრები, ComboBox FileDialog, TabCtrl, ControlBar, Button, ListBox, ListCtrl და სხვა **AcUI** და **AdUI** კლასების და ობიექტების იერარქია შემდეგნაირად გამოიყურება:



პრეპროცესორის სისტემის ObjectARX ობიექტებით დაპროგრამების მეთოდის დამუშავება

წინა თავში აღწერილი ObjectARX კლასების და ობიექტების ერთობლიობიდან წინამდებარე დისერტაციაში, პრეპროცესორის შესაქმნელად გამოყენებული იქნა შემდეგი კლასები და ობიექტები:

ცხრილი №1:

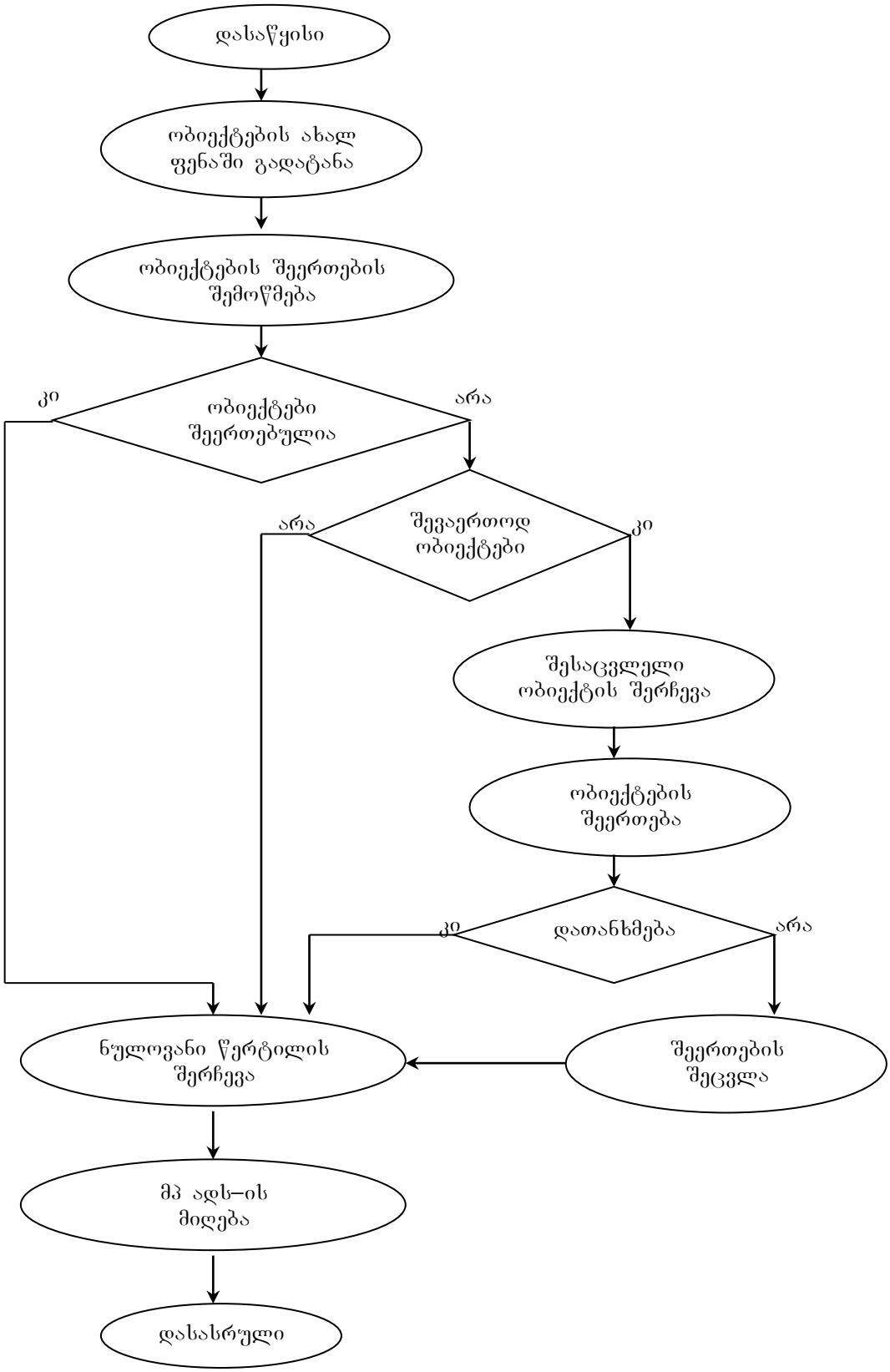
კლასი	კლასის აღწერა	თვისებები
acdbHostApplicationServices()	კლასი გამოიყენება AutoCAD_ის ძირითადი სერვისებით მანიპულირებისათვის	getSymbolTable()
AcDbLayerTable	გამოიყენება ფენების (Layer) ბაზით მანიპულირებისათვის	Open() Add() Close()
AcDbLayerTableRecord	გამოიყენება ფენების (Layer) ბაზის ჩანაწერებით მანიპულირებისათვის	setName() SetColor()
acedEntSel	გამოიყენება ობიექტის მოსანიშნად	Prompt
AcDbObjectId	გამოიყენება ობიექტის ID–ს განსაზღვრისათვის	GetObjectId()
AcDbEntity	გამოიყენება AutoCAD–ის ობიექტების მონაცემთა ბაზის ობიექტის აღწერისათვის	Open() Close() isKindOf() setLayer() setcolor() highlight() unhighlight() getTransformedCopy()

ცხრილი №1 გაგრძელება:

კლასი	კლასის აღწერა	თვისებები
AcDbCurve	გამოიყენება AutoCAD–ის "Curve" ტიპის ობიექტებით მანიპულირებისათვის	getStartPoint() getEndPoint() setStartPoint() setEndPoint()
AcDbLine	გამოიყენება AutoCAD–ის ობიექტით "ხაზი (Line)" მანიპულირებისათვის	StartPoint() EndPoint() setStartPoint() setEndPoint()
AcDbArc	გამოიყენება AutoCAD–ის ობიექტით "რკალი (Arc)" მანიპულირებისათვის	getStartPoint() getEndPoint() setStartPoint() setEndPoint() startAngle() EndAngle() center() radius()
AcDbPoint	გამოიყენება AutoCAD–ის ობიექტით "წერტილი (Point)" მანიპულირებისათვის	Position() SetPosition() distanceTo()
acedCommand()	ამ კლასის გამოყენებით ObjectARX–ის ბრძანებები გადაეცემა პირდაპირ AutoCAD–ის ბრძანებათა ხაზს (CommandLine)	argument argument type
acutPrintf()	ამ კლასის გამოყენებით შესაძლებელია AutoCAD–ის ბრძანებათა ხაზში ტექსტური შეტყობინებების გამოტანა	Prompt

“პრეპროცესორი”–ს მუშაობის ალგორითმი

პრეპროცესორის მუშაობის ალგორითმი ვიზუალურად შემდეგნაირად გამოიყურება:



I. საკონსტრუქტორო ნახაზიდან ოპერაციული ნახაზის მისაღებად AutoCAD_ის ბრძანებათა ხაზში მომხმარებელი კრიფავს ბრძანებას “MYSELECT” (ან პრეპროცესორის თულბარის შესაბამის ღილაკზე დაჭერით იძახებს ამ ბრძანებას). AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Select Objects to Copy in New Layer” (მონიშნეთ ობიექტები ახალ ფენაში გადატანის მიზნით). მომხმარებელი ნიშნავს სასურველ ობიექტებს რომლებიც, მონიშვნის დასრულების შემდეგ, გადაიტანება ახალ ფენაში “Test_Layer”.

II. ოპერაციული ნახაზის შემოწმების მიზნით AutoCAD_ის ბრძანებათა ხაზში მომხმარებელი კრიფავს ბრძანებას “MYCHECK” (ან პრეპროცესორის თულბარის შესაბამის ღილაკზე დაჭერით იძახებს ამ ბრძანებას). AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Select First Object” (მონიშნეთ პირველი ობიექტი). იმ ორი ობიექტიდან, რომელთა შეერთების შემოწმება ხდება, მომხმარებელი ნიშნავს ერთერთს. AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Select Second Object” (მონიშნეთ მეორე ობიექტი). მომხმარებელი ნიშნავს მეორე ობიექტს. თუ მონიშნული ეს ორი ობიექტი ერთმანეთთან შეერთებულია AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Objects Are Connected” (ობიექტები შეერთებულია). თუ ეს ორი ობიექტი ერთმანეთთან არ არის შეერთებული AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Objects Are NOT Connected, Do You want to Connect Them?[Y/N]” (ობიექტები არ არის შეერთებული, გსურთ მათი შეერთება?[კი/არა]). თუ მომხმარებელს არ სურს ობიექტების შეერთება აჭერს ღილაკს “N”, პროგრამა ასრულებს მუშაობას. თუ მომხმარებელს სურს ობიექტების შეერთება, აჭერს ღილაკს “Y”, AutoCAD_ის ბრძანებათა ხაზში გამოდის

შეტყობინება: “Select Object To Change” (რომელი ობიექტი შეიცვალოს), მომხმარებელი ნიშნავს ობიექტს რომლის შეცვლაც სურს, პრეპროცესორი ახდენს ამ ობიექტის დაკავშირებას მეორე ობიექტის საწყის ან საბოლოო წერტილთან, AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Objects Successfully Connected,Do you whant to Change Conection?[Y/N]” (ობიექტები შეერთებულია, გსურთ შეერთების შეცვლა? [კი/არა]), თუ მომხმარებელს აკმაყოფილებს შეერთების ფორმა ის აჭერს რილაკს “N”, პროგრამა ასრულებს მუშაობას. თუ მომხმარებელს არ აკმაყოფილებს შეერთების ფორმა ის აჭერს რილაკს “Y”, პრეპროცესორი ცვლის ობიექტების შეერთების წერტილს, პროგრამა ასრულებს მუშაობას.

III. კოორდინატთა სისტემის ტრანსფორმაციის მიზნით AutoCAD_ის ბრძანებათა ხაზში მომხმარებელი კრიფავს ბრძანებას “MYZERO” (ან პრეპროცესორის თულბარის შესაბამის ღილაკზე დაჭერით იძახებს ამ ბრძანებას). AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Select ZERO Point” (მონიშნეთ ნულოვანი წერტილი). მომხმარებელი ნიშნავს წერტილს რომელშიც გადაიტანება კოორდინატთა სისტემის ნულოვანი წერტილი.

IV. საყრდენი წერტილების კოორდინატების მიღების მიზნით AutoCAD_ის ბრძანებათა ხაზში მომხმარებელი კრიფავს ბრძანებას “MYPRINT” (ან პრეპროცესორის თულბარის შესაბამის ღილაკზე დაჭერით იძახებს ამ ბრძანებას). AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “Select Contour” (მონიშნეთ კონტური). მონიშვნის დასრულების შემდეგ AutoCAD_ის ბრძანებათა ხაზში გამოდის შეტყობინება: “View Results in Result.txt” (იხილეთ შედეგი Result.txt ფაილში). შედეგები იწერება Result.txt ფაილში შემდეგი სახით (მაგალითი):

ცხრილი №2.

N	X	Z	R
1	0	0	
2	5.00	5.00	
3	10.00	12.00	20
4	20.50	25.50	
5	35.00	37.50	5
6	40.05	50.10	
7	60.00	65.50	

პროგრამის ლისტინგი

```
#include "StdAfx.h"
#include "StdArx.h"
#include "dbapserv.h"
#include "dbid.h"
#include "math.h"
#include "ymath.h"
#include <iostream.h>
#include <fstream.h>
#include <dbcolor.h>
#include <dbidmap.h>



void createNewLayer(const char *layerName)
{
AcDbLayerTable *pLayerTable;
acdbHostApplicationServices()->workingDatabase()->getSymbolTable(pLayerTable,
AcDb::kForWrite);

AcDbLayerTableRecord *pLayerTableRecord = new AcDbLayerTableRecord;
pLayerTableRecord->setName(layerName);

AcCmColor color;
color.setColorIndex(1);
pLayerTableRecord->setColor(color);

pLayerTable->add(pLayerTableRecord);
pLayerTable->close();
pLayerTableRecord->close();
}



void CopyingObjects(const char *layerName)
{

m:

ads_name Obj1 ;

ads_point point1 ;

if ( acedEntSel ("\nSelect Object(s) To Copy In New Layer: ", Obj1, point1) != RTNORM )
return ;
```

```
AcDbObjectId id1 ;

acdbGetObjectId (id1, Obj1) ;

AcDbEntity *pEnt1;

acdbOpenObject(pEnt1, id1, AcDb::kForRead);


if (pEnt1->isKindOf(AcDbLine::desc()))
{
goto a;
}
else if (pEnt1->isKindOf(AcDbArc::desc()))
{

goto a;
}
else
{
acutPrintf("Selected Object Is NOT Line or Arc");

pEnt1->close();
goto m;
}
a:
AcDbEntity *pEnt2;
AcGeMatrix3d Matr;
pEnt1->getTransformedCopy(Matr,pEnt2);

AcDbBlockTable *pBlockTable;
acdbHostApplicationServices()->workingDatabase()->getSymbolTable(pBlockTable,
AcDb::kForRead);

AcDbBlockTableRecord *pBlockTableRecord;
pBlockTable->getAt(ACDB_MODEL_SPACE, pBlockTableRecord,AcDb::kForWrite);
pBlockTable->close();

AcDbObjectId Id;
pBlockTableRecord->appendAcDbEntity(Id, pEnt2);
pBlockTableRecord->close();

pEnt2->setLayer(layerName);
pEnt2->setColorIndex(1);
pEnt1->close();
pEnt2->close();

goto m;
```

```
}


void HideLayers()
{
acedCommand(RTSTR, "layer",RTSTR, "Set",RTSTR, "TEST_LAYER",RTSTR, "",0);
acedCommand(RTSTR, "layer",RTSTR, "OFF",RTSTR, "*",RTSTR, "yes",RTSTR, "",0);
acedCommand(RTSTR, "layer",RTSTR, "ON",RTSTR, "TEST_LAYER",RTSTR, "",0);
acedCommand(RTSTR,"zoom",RTSTR,"e",0);

}


void chek()
{

ads_name Object1 ;
ads_point pt1 ;
if ( acedEntSel ("\nSelect First Object: ", Object1, pt1) != RTNORM )
return ;

AcDbObjectId idd1 ;
acdbGetObjectId (idd1, Object1) ;

AcDbCurve *pEntt1;
acdbOpenObject(pEntt1, idd1, AcDb::kForRead);

pEntt1->highlight();


ads_name Object2 ;
ads_point pt2 ;
if ( acedEntSel ("\nSelect Second Object: ", Object2, pt2) != RTNORM )
return ;

AcDbObjectId idd2 ;
acdbGetObjectId (idd2, Object2) ;

AcDbCurve *pEntt2;
acdbOpenObject(pEntt2, idd2, AcDb::kForRead);

pEntt2->highlight();




AcGePoint3d st1;
AcGePoint3d end1;
AcGePoint3d st2;
AcGePoint3d end2;
```

```
pEntt1->getStartPoint(st1);
pEntt1->getEndPoint(end1);
pEntt2->getStartPoint(st2);
pEntt2->getEndPoint(end2);



if (st1==st2 || st1==end2 || end1==st2 || end1==end2)

{

acutPrintf("\nObjects Are Conected\n");

pEntt1->unhighlight();
pEntt2->unhighlight();

pEntt1->close();
pEntt2->close();
}
else

{

acutPrintf("\nObjects Are Not Conected\n");
char Answer;
acedInitGet (RSG_NONULL,"y n") ;
acedGetString(0,"\nDo You Want To Connect Them?[Y/N]", &Answer);



if (Answer=='y'|| Answer=='Y')
{
pEntt1->unhighlight();
pEntt2->unhighlight();
pEntt1->close();
pEntt2->close();

ads_name Object ;
ads_point pt ;
acedEntSel ("\nSelect Which Object to change: ", Object, pt) ;

AcDbObjectId id3 ;
acdbGetObjectId (id3, Object) ;

AcDbCurve *ChEnt;
acdbOpenObject(ChEnt, id3, AcDb::kForRead);

if (ChEnt->isKindOf(AcDbLine::desc()))
```

```
{
ChEnt->close();

////////////// certilebis dadgena//////////
double Len1,Len2,Len3,Len4;

Len1=st1.distanceTo(st2);
Len2=st1.distanceTo(end2);
Len3=end1.distanceTo(st2);
Len4=end1.distanceTo(end2);


double min1,min2,minimum;
min1=min(Len1,Len2);
min2=min(Len3,Len4);
minimum=min(min1,min2);


AcGePoint3d Point1;
AcGePoint3d Point2;



if (minimum==Len1)
{
Point1=st1;
Point2=st2;

}
else if (minimum==Len2)
{
Point1=st1;
Point2=end2;

}
else if (minimum==Len3)
{
Point1=end1;
Point2=st2;

}
else if (minimum==Len4)
{
Point1=end1;
Point2=end2;

}
```

```
AcDbLine *Line1;
AcDbObjectId id4;
id4=id3;

acdbOpenObject(Line1, id4, AcDb::kForRead);

AcGePoint3d st;
AcGePoint3d end;
Line1->getStartPoint(st);
Line1->getEndPoint(end);

if (st==Point1)
{
Line1->upgradeOpen();
Line1->setStartPoint(Point2);
Line1->close();
acutPrintf("\nObjects sucsesfully connected");
}
if (st==Point2)
{
Line1->upgradeOpen();
Line1->setStartPoint(Point1);
Line1->close();
acutPrintf("\nObjects sucsesfully connected");
}
if (end==Point1)
{
Line1->upgradeOpen();
Line1->setEndPoint(Point2);
Line1->close();
acutPrintf("\nObjects sucsesfully connected");
}
if (end==Point2)
{
Line1->upgradeOpen();
Line1->setEndPoint(Point1);
Line1->close();
acutPrintf("\nObjects sucsesfully connected");
}


}


else if (ChEnt->isKindOf(AcDbArc::desc()))
{
ChEnt->close();
```

```
////////////// certilebis dadgena//////////
double Len1,Len2,Len3,Len4;

Len1=st1.distanceTo(st2);
Len2=st1.distanceTo(end2);
Len3=end1.distanceTo(st2);
Len4=end1.distanceTo(end2);


double min1,min2,minimum;
min1=min(Len1,Len2);
min2=min(Len3,Len4);
minimum=min(min1,min2);


AcGePoint3d Point1;
AcGePoint3d Point2;



if (minimum==Len1)
{
Point1=st1;
Point2=st2;

}
else if (minimum==Len2)
{
Point1=st1;
Point2=end2;

}
else if (minimum==Len3)
{
Point1=end1;
Point2=st2;

}
else if (minimum==Len4)
{
Point1=end1;
Point2=end2;

}
```

```
AcGePoint3d MyPoint;

AcDbArc *Arc1;
AcDbObjectId id4;
id4=id3;

acdbOpenObject(Arc1, id4, AcDb::kForRead);

AcGePoint3d st;
AcGePoint3d end;
Arc1->getStartPoint(st);
Arc1->getEndPoint(end);

AcGePoint3d aaa;

if  (st==Point1)
{
MyPoint=Point2;
aaa=end;
}
if (st==Point2)
{
MyPoint=Point1;
aaa=end;
}
if (end==Point1)
{
MyPoint=Point2;
aaa=st;
}
if (end==Point2)
{
MyPoint=Point1;
aaa=st;
}



AcGePoint3d centre;
double stA, endA;
double radius;


centre=Arc1->center();
stA=Arc1->startAngle();
endA=Arc1->endAngle();
radius=Arc1->radius();
Arc1->upgradeOpen();
```

```
Arc1->erase();
Arc1->close();

double mid_x, mid_y, mid_z;
double centre_x, centre_y, centre_z;



centre_x=centre[0];
centre_y=centre[1];
centre_z=centre[2];

if (centre_y>st[1])
{
mid_x=centre_x+cos((stA+endA)/2)*radius;
mid_y=centre_y-sin((stA+endA)/2)*radius;
mid_z=0;
}
else
{
mid_x=centre_x+cos((stA+endA)/2)*radius;
mid_y=centre_y+sin((stA+endA)/2)*radius;
mid_z=0;
}


ads_point mid;
mid[0]=mid_x;
mid[1]=mid_y;
mid[2]=mid_z;

double Point1_x,Point1_y,Point1_z;
Point1_x=Point1[0];
Point1_y=Point1[1];
Point1_z=Point1[2];


ads_point bolo;
bolo[0]=aaa[0];
bolo[1]=aaa[1];
bolo[2]=aaa[2];

ads_point stbolo;
stbolo[0]=MyPoint[0];
stbolo[1]=MyPoint[1];
stbolo[2]=MyPoint[2];

acedCommand(RTSTR,"Arc",RT3DPOINT,stbolo, RT3DPOINT,mid, RT3DPOINT,bolo,0);
acutPrintf("\nObjects sucsesfully connected");
```

```
}

else
{
ChEnt->close();
acutPrintf("\n it is not a Line Or Arc");
}

//////////////////dalshe///////////////////////////////

}
else if (Answer=='n'|| Answer=='N')
{

pEntt1->unhighlight();
pEntt2->unhighlight();
pEntt1->close();
pEntt2->close();
return;
}
else
{
pEntt1->unhighlight();
pEntt2->unhighlight();
pEntt1->close();
pEntt2->close();
return;
}

}

AcDbObjectId::setNull;
}



void zero()
{
ads_point zero;
zero[0]=0;
zero[1]=0;
zero[2]=0;

ads_point userpoint;

acedGetPoint(0,"Select ZERO Point",userpoint);

acedCommand(RTSTR,"move",RTSTR,"all",RTSTR,"",RT3DPOINT,userpoint,RT3DPOINT,zero
,0);
acedCommand(RTSTR,"zoom",RTSTR,"e",0);
```

```
}


void print()
{


double x,y,r;
int i=0;

ofstream out("result.txt");
out <<  "N       X        Z        R" << endl;
out <<  "---------------------------------" << endl;

AcGePoint3d pt1;
AcGePoint3d pt2;


ads_name Object ;
ads_point pt ;
AcDbObjectId id ;
AcDbCurve *pEnt;

if (acedEntSel ("\nSelect Path ", Object, pt)!=RTNORM)
{
acutPrintf("\nNC List Created Sucsessfilly.\nView Rezult.txt");

return;
}


acdbGetObjectId (id, Object) ;
acdbOpenObject(pEnt, id, AcDb::kForRead);


AcGePoint3d point1;
AcGePoint3d point2;
pEnt->getStartPoint(point1);
pEnt->getEndPoint(point2);
pEnt->highlight();
pEnt->close();
pt1=point1;
pt2=point2;

if (pEnt->isKindOf(AcDbArc::desc()))
{

AcDbArc *pEnt;
r=pEnt->radius();
acutPrintf("%d",r);
```

```
pEnt->close();
}


start:

if (acedEntSel ("\nSelect Path ", Object, pt)!=RTNORM)

{

acutPrintf("\nNC List Created Sucsessfilly.\nView Rezult.txt");
pEnt->unhighlight();
return;
}


pEnt->unhighlight();
acdbGetObjectId (id, Object) ;


acdbOpenObject(pEnt, id, AcDb::kForRead);

AcGePoint3d st;
AcGePoint3d en;
pEnt->getStartPoint(st);
pEnt->getEndPoint(en);
pEnt->highlight();
pEnt->close();


if (point1==st || point1==en)
{
x=point2[0];
y=point2[1];
i++;

y=int(y*100);
y=y/100;

x=int(x*100);
x=x/100;
out <<i<<"\t"<<y<<"\t "<<x<< endl;
x=point1[0];
y=point1[1];
i++;

y=int(y*100);
y=y/100;

x=int(x*100);
```

```
x=x/100;
out <<i<<"\t"<<y<<"\t "<<x<< endl;
}
else if (point2==st || point2==en)
{
x=point1[0];
y=point1[1];
i++;

y=int(y*100);
y=y/100;

x=int(x*100);
x=x/100;
out <<i<<"\t"<<y<<"\t "<<x<< endl;
x=point2[0];
y=point2[1];
i++;

y=int(y*100);
y=y/100;

x=int(x*100);
x=x/100;
out <<i<<"\t"<<y<<"\t "<<x<< endl;
}

if (st==pt1 || st==pt2)
{
x=en[0];
y=en[1];
}
else if (en==pt1 || en==pt2)
{
x=st[0];
y=st[1];
}

pt1=st;
pt2=en;

y=int(y*100);
y=y/100;

x=int(x*100);
x=x/100;
i++;
out <<int(i)<<"\t"<<y<<"\t "<<int(x)<< endl;

goto start;
```

```
out.close();
}
```