UNIVERSITATEA „POLITEHNICA” TIMIŞOARA

SISTEM INFORMATIC EDUCAȚIONAL PENTRU GEOMETRIE

TEZĂ DE DOCTORAT

DOCTORAND:
Anca Elena IORDAN

CONDUCĂTOR ŞTIINȚIFIC:
Prof.univ.dr.ing. George SAVII

2009

CUPRINS

LISTA FIGURILOR ŞI TABELELOR

INTRODUCERE

Prezenta teză îşi propune să reliefeze contribuţiile proprii rezultate în activitatea de cercetare în cadrul programului de doctorat cu tema *"Sistem informatic educaţional pentru geometrie"*.

În realizarea sistemului informatic interactiv destinat studiului geometriei s-a urmărit atingerea următoarelor scopuri:

- prezentarea noţiunilor teoretice şi a principalelor rezultate;
- prezentatea interactivă de aplicaţii pentru fiecare subdomeniu solicitat;
- realizarea de desene exacte prin înlocuirea creionului şi a riglei cu *mouse*-ul.

După proiectarea arhitecturii funcţionale a sistemului care să permită susţinerea activităţii didactice, este propus un model pentru interfaţa grafică care să satisfacă necesităţile specifice unui mediu didactic din punctul de vedere al profesorilor, studenţilor şi elevilor.

Proiectarea interfeţelor grafice utilizator reprezintă faza cea mai importantă a procesului de implementare a unui sistem interactiv de instruire. Interfaţa cu utilizatorul trebuie să furnizeze toate facilităţile necesare unui student sau unui elev pentru a putea naviga în aplicaţie intuitiv şi cât mai transparent posibil. Proiectarea concentrată asupra utilizatorului impune interfeţei caracteristicile care permit utilizatorului să controleze procesul de instruire. O interfaţă utilizabilă trebuie studiată, realizată şi testată iterativ pentru a putea maximiza eficienţa şi minimiza timpul necesar proceselor de predare şi instruire.

1.1 Obiectivele tezei

În realizarea sistemului informatic interactiv destinat procesului de asimilare a cunoştinţelor din domeniul geometriei s-a urmărit îndeplinirea următoarelor obiective:

- Implemetarea unui program proiectat pentru achiziţia de noi cunoştinţe ce vor acoperi un domeniu larg al geometriei cum ar fi: triunghiul, patrulaterul, vectori şi cuadrice.

- Realizarea exactă a desenelor în plan şi în spaţiu. Interfaţa grafică corespunzătoare acestui obiectiv va cuprinde o bară de meniuri şi o bară de butoane ce vor oferi instrumentele necesare pentru obţinerea exactă a desenului dorit, precum şi suprafaţa de desenare. Dintre cele mai importante opţiuni amintim:
  - desenarea punctelor libere sau a punctelor cu anumite proprietăţi, cum ar fi: mijlocul unui segment, centrul de greutate al unui triunghi, ortocentrul unui triunghi, centrul cercului lui Euler, punctul lui Lemoine, punctul lui Gergonne, intersecţia unei dreptei cu un plan, centrul unei cuadrice etc.;
  - desenarea dreptelor determinate de două puncte, a dreptelor determinate de un punct şi pantă, a dreptelor determinate de două plane, a dreptelor determinate de un punct şi un vector director liber sau a dreptelor ce îndeplinesc anumite condiţii, de exemplu: paralela şi perpendiculara printr-un punct la o dreaptă, mediatoarea unui segment, tangenta şi normala într-un punct la o conică, normala într-un punct la o cuadrică etc.;

- desenarea semidreptelor determinate de două puncte sau a semidreptelor determinate de anumite proprietăți;
- desenarea segmentelor determinate de două puncte sau a segmentelor a căror extremități sunt puncte particulare, cum ar fi: mediana şi simediana într-un punct corespunzătoare unui vârf al triunghiului, diagonala unui patrulater, linia mijlocie într-un trapez, diametrul unui cerc, coarda ce subîntinde un arc de cerc etc.;
- desenarea vectorilor determinați de origine şi extremitate, a vectorilor de poziție, a vectorului director al unei drepte, a vectorului normal al unui plan;
- desenarea unghiului determinat de trei puncte sau a unghiului determinat de două semidrepte;
- desenarea planului determinat de trei puncte, a planului determinat de două drepte, a planului determinat de cei patru coeficienți sau a unui plan determinat prin anumite proprietăți, de exemplu: planul perpendicular pe o dreaptă determinat de un punct, planul tangent la o cuadrică într-un punct specificat etc;
- desenarea poligoanelor oarecare şi a poligoanelor regulate, dar şi a unor poligoane speciale, cum ar fi: triunghiul ortic, triunghiul median etc;
- desenarea conicelor determinate de cinci puncte şi a conicelor particulare: elipsă, cerc, hiperbolă şi parabolă;
- desenarea cuadricelor: elipsoid, hiperboloid, paraboloid, sfera;
- desenarea pseudosferei şi a elicoidului;
- aplicarea de transformări geometrice: izometrii (simetrie, translație, rotație), omotetie, inversiune.

▪ Proiectarea sistemului interactiv prin identificarea celor mai importante concepte corespunzătoare elementelor geometrice în plan şi în spațiu şi implementarea acestora într-un mod eficient.

▪ Implementarea unor algoritmi de intersecție şi de tangență care vor contribui la rapiditatea şi exactitatea reprezentării grafice a elementelor geometrice dorite.

1.2 Structura tezei

În acest context, structura logică pe baza căreia a fost construită lucrarea de față este următoarea:

În **capitolul doi** este prezentat stadiul actual în cercetările privind sistemele de instruire asistată de calculator la geometrie. Capitolul trece în revistă domeniul interdisciplinar de cercetare. Se face un scurt istoric al instruirii asitate de calculator, apoi este analizat rolul profesorului în învățământul actual. Este prezentat rolul programelor de instruire asistată de calculator şi este propusă o clasificare a acestora din punct de vedere pedagogic. În continuare sunt analizate considerentele psiho-pedagogice ale utilizării calculatorului în procesul de studiere al matematicii. În acest capitol sunt prezentate contribuțile la realizarea unor programe educaționale destinate studierii unor capitole ale matematicii, fizicii şi informaticii, iar, în încheiere, sunt prezentate sistemele de instruire existente pentru geometrie, sintetizând avantajele şi dezavantajele fiecărui sistem prezentat.

În **capitolul trei** este descris sistemul informatic interactiv realizat prin prezentarea celor trei etape esențiale: analiză, proiectare şi implementare. Prin reprezentarea diagramelor corespunzătoare celor trei faze: analiză, proiectare şi implementare sistemul informatic interactiv va fi descris într-o manieră clară şi concisă. Utilizarea limbajului de modelare UML în realizarea diagramelor este caracterizată de rigoare sintactică, semantică bogată şi suport pentru modelarea vizuală.

În **capitolul patru** este realizată o justificare a limbajului de programare ales şi sunt prezentate structurile de date optimizate utilizate pentru geometria dinamică. Tot în acest capitol este prezentată interfața grafică a sistemului interactiv, interfața ce va permite manipularea sistemului într-o manieră foarte avantajoasă şi simplă. Vor fi prezentate caracteristicile ferestrelor corespunzătoare celor două cazuri de realizare de construcții geometrice, în plan şi în spațiu, precum şi a celor patru cazuri de prezentare de noțiuni şi rezultate.

În **capitolul cinci** se vor defini mulțimea operațiilor de bază specifice unui sistem dinamic pentru geometrie şi vor fi descrise explicit cum pot fi efectuate în coordonate carteziene calculele necesare operațiilor de bază.

Ultimul capitol al lucrării este consacrat concluziilor acestei teze şi perspectivelor.

1.3 Concluzii

Obiectivul central al cercetării realizate în cadrul temei tezei de doctorat intitulate**:** *"Sistem informatic educațional pentru geometrie"* este proiectarea şi implementarea unui mediu dinamic interactiv care să conducă utilizatorul la obținerea unei experiențe în înțelegerea şi stăpânirea de cunoştințe din domeniul geometriei şi să ofere accesul comod şi eficient la informațiile şi cunoştințele cele mai noi.

Tema tratată în cadrul tezei este de mare actualitate, geometria fiind o componentă importantă în formarea tinerilor matematicieni, ingineri sau arhitecți. Aplicația informatică a tezei de doctorat se înscrie în această problematică majoră şi de certă actualitate, pe care o abordează într-o manieră nouă, prin prisma tehnologiilor educaționale moderne. Prin utilizarea metodelor specifice învățământului la distanță, teza de doctorat se constituie într-o abordare multidisciplinară.

STADIUL ACTUAL ÎN CERCETĂRILE PRIVIND SISTEMELE DE INSTRUIRE ASISTATĂ DE CALCULATOR LA GEOMETRIE

2.1 Instruirea asistată de calculator

Învățământul asistat de calculator [1] este un termen general utilizat pentru a defini toate aplicațiile sistemelor de calcul în unitățile şi activitățile de învățământ. Cea mai răspândită dintre aplicații o constituie instruirea asistată de calculator în care cei ce învață comunică interactiv cu sistemul de calcul, utilizând un sistem de programe destinat învățării în cele mai diverse domenii. De obicei sistemul de programe este realizat astfel încât să prezinte cursantului o cantitate de informație iar apoi, alternativ, să testeze modul de înțelegere şi însuşire a respectivei informații. Sistemul de programe permite contabilizarea răspunsurilor corecte şi eronate pentru fiecare cursant. Elaborarea unor astfel de sisteme necesită eforturi serioase de programare, esențiale fiind problemele de dozare a informației şi de formulare a întrebărilor de verificare.

Calculatorul oferă posibilități reale de individualizare a instruirii. El nu este doar un mijloc de transmitere a informației ci poate oferi programe de învățare adaptate conduitei şi cunoştințelor elevului. Colaborarea dintre informaticieni, constructori de calculatoare şi specialişti din domeniul instrucției şi educației a permis inițierea unor programe concrete, privind folosirea calculatoarelor în procesul de învățământ. Realizarea unei metodologii care să facă eficientă asistarea procesului de învățământ cu calculatorul a solicitat folosirea instrumentelor psiho-pedagogiei.

Conceptul de asistare a procesului de învățământ cu calculatorul include:

- predarea unor lecții de comunicare de noi cunoştințe;
- aplicarea, consolidarea, sistematizarea noilor cunoştințe;
- verificarea automată a unei lecții sau a unui grup de lecții;
- verificarea automată a unei discipline şcolare sau a unei anumite programe şcolare.

Se disting două moduri, nu neapărat exclusive, de intervenție a computerului în instruire:

- direct: când computerul îndeplineşte principala sarcină a profesorului, adică predarea;
- indirect: computerul funcționează ca manager al instruirii.

Intervenția directă a computerului se poate face printr-un *software* educațional şi este descrisă de termenul Instruire Asistată de Calculator. Intervenția indirectă constă în utilizarea computerului pentru controlul şi planificarea instruirii în care calculatorul preia o parte din sarcinile profesorului:

- prezintă elevului obiectivele de atins şi părțile componente ale cursului;
- atribuie sarcini de lucru specifice din manualul sau caietul de lucru asociat cursului respectiv;
- administrează teste pentru a determina progresul elevului în raport cu directivele prestabilite;
- înregistrează şi raportează rezultatele obținute la teste pentru elev sau profesor;
- prescrie, în funcție de rezultatele la un test diagnostic, ce secvență va studia în continuare un anumit elev.

Posibilitățile mediilor bazate pe computer în ceea ce priveşte tratamentul, înregistrarea şi regăsirea informației vor determina introducerea în practica pedagogică a situațiilor în care elevul va dobândi cunoştințe şi competențe în mod autonom, în conformitate cu interesele şi aspirațiile proprii, prin intermediul unor instrumente informatice.

Acceptând ideea civilizației informatice, trebuie admisă şi aceea a instruirii în spiritul interesului pentru informație, aceasta fiind privită ca resursa cea mai de preț a omenirii, neconsumabilă şi conştientizată doar de puțin timp ca resursă naturală. Deci informația a devenit un element al infrastructurii şi aceasta a determinat apariția unui fenomen important: comanda socială a societății impune tot mai mult însuşirea unei cât mai largi culturi generale informatice. Adică, luând în seamă necesitățile automatizării, robotizării, cibernetizării pe scară largă a proceselor economice rezultă imperativul familiarizării, încă de pe băncile şcolii cu modul de lucru şi cu facilitățile oferite de tehnica de automatizare.

Se diferențiază mai multe niveluri de asimilare a calculatorului în învățământ:

- nivelul inițierii şi acomodării (ciclul primar şi gimnazial);
- nivelul aprofundării şi exersării (ciclul liceal);
- nivelul dezvoltării de aplicații cu grad înalt de complexitate (ciclurile universitar şi postuniversitar).

2.2 Schimbarea rolului profesorului în învățământ

Foarte mult timp sistemul de învățământ a fost un sistem de învățământ centrat pe profesor. Profesorul era persoana care transforma conținuturile disciplinei într-o formă accesibilă înțelegerii elevilor. Instruirea programată, iar apoi şi instruirea computerizată a început să submineze treptat modelul tradițional de instruire: funcțiile profesorului au început să fie îndeplinite de maşini. Noua metodă de instruire - instruirea computerizată - promite a-şi lărgi domeniile de aplicare, asumându-şi treptat noi funcții ale profesorului [18,81].

Nenumăratele ore de utilizare practică a programelor computerizate de instruire au evidențiat multe aspecte ale relației: calculator-personalitate. Este demonstrat, că personalitatea profesorului are o mare însemnătate în formarea cunoştințelor primare. În primul rând, elevii doresc să vadă în profesor un ideal al lor, cult, inteligent, deştept, cu o anumită „pondere” în ştiință, societate. Când un aşa profesor iese în fața celor pe care îi învață, aceştia doresc să învețe de la el totul pentru ca să posede şi ei o gamă largă de cunoştințe, să fie buni specialişti, să ocupe o poziție anumită în societate.

În multiple lucrări de pedagogie [14,25,69,75] se menționează, că nu numai personalitatea profesorului joacă un rol primordial în formarea cunoştințelor, ci şi însăşi prezența lui în fața elevilor. În timpul lecției profesorul „emană o anumită energie”, care este asimilată de cei care îl audiază. Acest lucru a fost elocvent demonstrat prin următorul experiment:

La trei grupe de studenți li s-a propus să audieze un curs. În prima grupă acest curs a fost susținut de către un profesor (în persoană) nu prea calificat; următoarei grupe i s-a propus să audieze acest curs imprimat la magnetofon de un profesor ilustru; iar grupei a treia - să studieze acelaşi curs de la un videomagnetofon, unde acesta era înscris la fel de către un profesor ilustru. Rezultatele experimentului au dovedit că cele mai bune performanțe au fost atinse în cazul primei grupe, unde a lucrat însuşi profesorul.

Specialiştii psihologi demonstrează că calculatorul poate să ia asupra sa, în primul rând, organizarea şi optimizarea procesului instructiv-educativ, efectuarea procesului instructiv la nivelul formării aptitudinilor si deprinderilor, organizarea evaluării cunoştințelor, organizarea demonstrațiilor, modelarea şi simularea diferitelor procese reale, automatizarea îndeplinirii lucrărilor practice şi de laborator. Cu mult succes calculatorul poate îndeplini, prin intermediul

programelor sale, rolul de mediator. Multiple studii demonstrează că instruirea computerizată ridică calitatea cunoştinţelor, reduce timpul necesar pentru acumularea acestor cunoştinţe, iar cunoştinţele deja formate prin instruirea computerizată sunt mai operative şi mai trainice. Este confirmat, deci, faptul că un calculator operează mult mai bine în rolul de mediator decât în rol de formator al cunoştinţelor primare. Prezintă interes ideea potrivit căreia calculatorul este în stare să formalizeze tipul de gândire a elevului şi să stimuleze creşterea capacităţilor lui cognitive.

În aceste condiţii rolul profesorului în procesul de instruire se schimbă, în mod evident, simţitor [19,22]. Deci, un profesor contemporan în condiţiile utilizării instruirii asistate de calculator rămâne să îndeplinească următoarele funcţii:

- de a motiva învăţarea;
- de a forma la elevi cunoştinţele primare;
- de a deţine rolul de manager al educaţiei;
- de a fi moderator în orientarea corectă a utilizării calculatorului în procesul de învăţământ;
- de a fi facilitator al învăţării;
- de a dirija rezolvarea problemelor nestandarde;
- de a forma imaginea corectă a unei persoane de succes în conştiinţa elevilor.

Una dintre cele mai frecvent aplicate şi mai eficiente metode de utilizare a calculatorului în instruire este metoda instruirii individuale. Utilizarea programaturii cu orientarea spre această metodă duce la individualizarea maximală a învăţării, ceea ce reprezintă unul din scopurile de bază ale învăţământului contemporan. Dar, din alt punct de vedere, elevul, deprins să lucreze individual cu calculatorul va pierde deprinderile de lucru în grup şi de comunicare, se va închide în sine sau se va limita la comunicarea cu calculatorul, cea ce poate aduce la dependenţa de calculator, descrisă mai sus.

În ultimii ani, în legătură cu creşterea volumului de informaţie şi cu complicarea programelor de instruire accentele în procesul de predare-învăţare-evaluare se pun pe studiul independent al materiei de program sau suplimentare şi autoevaluare. Recunoscând posibilităţile calculatorului în acest domeniu, menţionăm, că ultimele studii psihologice şi pedagogice arată că majoritatea elevilor nu sunt în stare şi nu doresc să-şi planifice activitatea de studiu cu perspectivă îndelungată. Profesorul are un rol foarte important în stabilirea succesiunii şi volumului de informaţie studiat, în alegerea diferitelor exemple, probleme, în organizarea controlului asupra calităţii instruirii, în stabilirea *feed-back*-ului etc. În afară de aceasta, profesorul are funcţia de a urmări ca procesul de comunicare cu calculatorul să nu înlocuiască procesul de comunicare în colectivul clasei.

Şcoala tradiţională de multe ori ducea lipsă de aşa tipuri de comunicare între elevi precum cercurile pe interese, comunicarea dintre elevii de diferite vârste. Sfera de comunicare a unui elev se limita la familie, rude, colegii de clasă şi copiii din ogradă sau mahala. Calculatorul propune noi posibilităţi de comunicare: teleconferinţa, *chat*-ul, comunicarea prin *email*, care au menirea de a grupa utilizatorii după interese din diferite localităţi şi chiar ţări. Asupra rezolvării unei probleme poate lucra un grup de elevi, făcând schimb de rezultate şi discutând lucrul de mai departe asupra problemei.

Dar, trebuie de fixat, că însăşi programul de instruire şi celelalte posibilităţi ale calculatorului sunt doar un mijloc pentru rezolvarea unui cerc anumit de probleme în limitele unui sistem de instruire deja organizat, unde fiecare elev este implicat în instruirea activă cu evaluarea obiectivă a rezultatelor lui.

2.3 Influența calculatorului asupra perceperii informației

Legitățile didacticii generale stipulează că există diferite forme de influență asupra elevului şi diferite forme de percepere de către el a materiei de studiu: vizuale, audio şi audio-vizuale. Procesul de percepere a realității este un proces destul de complex şi constă, în esență, din formarea unei imagini senzoriale a obiectului. Perceperea include conştientizarea obiectelor, bazată pe includerea imaginii primite în sistemul de cunoştințe deja existent.

În procesul alegerii materiei care se va studia şi metodelor didactice folosite, profesorul trebuie să se bazeze pe principiul accesibilității. Utilizarea posibilităților multimedia a calculatoarelor contemporane permite de a stabili metode noi de studiere a materiei de program, apare posibilitatea explicării noțiunilor complicate într-un limbaj mai accesibil.

Arta de a preda constă nu numai în strictețea ştiințifică de expunere a materiei de studiu, dar şi în includerea în această activitate a tuturor formelor de influență asupra elevului. Utilizarea tehnologiilor multimedia în programul de instruire permite unificarea acestor forme şi oferă posibilitatea elevului de a alege automat forma de percepere comodă pentru sine [101]. Aceasta este deosebit de important în legătură cu faptul că, în dependență de particularitățile psihologice ale omului, viteza de percepere a informației de către el va fi mai mică decât viteza de lucru a calculatorului. O importanță deosebită o are şi caracterul informației transmise şi existența sau lipsa *feed-back*-ului din partea utilizatorului. Clasificând informația după acest principiu, obținem următoarele tipuri de informație:

- materie de studiu cu conținut de inițiere;
- material teoretic, bazat pe exemple reale, modelare matematică sau simulare;
- materie de evaluare;
- material teoretic abstract.

Viteza de însuşire a informației va fi minimală, dacă ea este din grupa a patra. Prelucrarea unei astfel de informații necesită eforturi foarte mari şi, prin urmare, se efectuează destul de lent. Transmiterea informației din grupa întâia şi a doua se efectuează mai rapid. Viteza de transmitere a informației în mare măsură depinde de nivelul de pregătire a instruitului. O importanță deosebită o are posibilitatea de încorporare în programul de instruire a componentei audio, adică a însoțirii parcurgerii programului prin vocea profesorului. Utilizând această componentă, profesorul primeşte posibilitatea de a dirija activitatea elevului prin observații audio pregătite anterior.

Există posibilitatea de a include în programul de instruire un sistem de asistență, organizat pe niveluri, ceea ce permite elevului de a se adresa după ajutor în momentul necesar, lucru efectiv irealizabil în condițiile unei clase de 20-30 de elevi cu un diferit nivel de pregătire în sistemul tradițional de instruire.

Tehnologiile multimedia au transformat calculatorul într-un interlocutor valoros şi au permis elevilor, fără a ieşi din clasa de studiu (din casă), să asiste la lecțiile diferiților savanți şi profesori emeriți, să comunice cu persoane, aflate în diferite țări, să aibă acces la diferite informații. Cu o singură activare a tastei *mouse*-ului elevii pot vizita o galerie artistică, citi originalele pentru scrierea unui referat la istorie sau vizualiza o informație pentru un profil îngust, care nu putea fi găsită în urmă cu cinci-zece ani. Pentru elevii care au ales un profil economic, există posibilitatea de a modela jocul la bursă, fără a risca cu mijloacele financiare proprii. Utilizarea posibilităților multimedia în programul de instruire computațional permite de a spori viteza procesului de asimilare a informației din contul interactivității şi ilustrativității.

Principiul ilustrativității este unul din principiile de bază ale didacticii, fiind introdus de Ian Comenschi [25]. În lucrările de pedagogie se menționează, că ilustrativitatea activizează în

foarte mare măsură procesul de cunoaştere a lumii reale. Tehnologiile multimedia întrec considerabil din acest punct de vedere mijloacele tradiționale, inclusiv mijloacele video şi cinema, deoarece pot fi utilizate atît dinamic, cît şi static.

În afară de aceasta, există programe specializate, care au posibilitatea să facă ilustrative obiectele ce nu sunt accesibile ochiului liber (de exemplu în cosmos sau în microlume). Ca urmare, se dezvoltă la maximum gândirea spațială şi perceperea imaginilor tridimensionale. Toate obiectele spațiale sunt tridimensionale şi perceperea lor de asemenea, dar proiecția lor pe retina ochiului este plană, bidimensională. Utilizarea la lecțiile de matematică a calculatorului permite să se reprezinte nu proiecțiile corpurilor, ci însăşi corpurile şi elevii pot percepe obiectele anume în forma în care ele există în lumea reală.

Unul dintre aspectele principale de utilizare a calculatorului la lecții este dezvoltarea gândirii creative a elevilor. Un mijloc optimal în acest caz este introducerea în mijloacele de instruire computaționale a elementelor de interactivitate. Termenul „interactivitate" semnifică „a interacționa, a influența unul asupra altuia". Această proprietate a tehnologiilor computaționale este absolut unicată în comparație cu televiziunea, prelegerile, cărțile, filmele instructive etc. Principiul interactivității a devenit un principiu euristic al instruirii asistate de calculator, deoarece este legat cu nişte caracteristici fundamentale a procesului de instruire: interacțiunea şi influența reciprocă a instruitului şi a celui ce instruieşte. În general, existența *feed-back*-ului este o trăsătură caracteristică oricărei programe calitative computaționale.

Astfel reieşind din premisele psiho-pedagogice a utilizării calculatorului în instruire şi a specificului aplicării lui putem face următoarele recomandări privind alegerea şi utilizarea informației de studiu, care poate fi inclusă în programele de instruire computaționale:

- materia de studiu trebuie bine structurată şi porționată;
- utilizarea la maximum a elementelor de interactivitate, în special, la rezolvarea problemelor;
- realizarea obligatorie a prezentărilor atractive şi demonstrative în programele standard, dar fără a face exces de ele, utilizându-le doar la momentele oportune;
- crearea unui sistem de asistență pe niveluri, încorporat în programele de instruire;
- utilizarea maximă a posibilităților multimedia a calculatorului;
- intercalarea lecțiilor tradiționale cu cele asistate de calculator.

2.4 Rolul programelor de instruire asistată de calculator

Învățământul realizează legătura dintre acțiunile didactice şi scopurile şi obiectivele ştiințific stabilite, prin elaborarea a noi metode şi prin asimilarea a noi mijloace, capabile să sporească randamentul şcolar, permițând elevilor să-şi însuşească sistemul cerut de cunoştințe şi tehnici de aplicare a acestora în condiții cât mai variate. Deoarece procesul didactic este în esență transmitere-însuşire de informații în scopul utilizării lor adecvate, ştiințele educației urmăresc şi asimilează realizări ale informaticii, grefate apoi pe sistemul propriu de principii, metode şi mijloace [72]. La intersecția informaticii cu didactica s-a profilat instruirea programată, iar o variantă deosebit de promițătoare a acesteia s-a obținut prin utilizarea calculatorului în scopuri didactice, rezultând instruirea asistată de calculator, care este o nouă dimensiune a instruirii şi care implică regândirea globală a procesului de învățământ, adaptarea lui la noile posibilități oferite de calculator.

Ultima generație de calculatoare, cu sisteme expert interactive, cu flexibilitate deosebită şi la un preț de cost accesibil, constituie un mijloc de învățământ cu totul deosebit, care pentru a fi utilizat la capacitatea oferită necesită o nouă formă de concepere şi organizare a procesului de învățământ. Ultimul conservă forma tradițională de organizare pe clase şi lecții, dar la care o bună parte din activitățile de rutină ale profesorului sunt suplinite, cu eficiență sporită, de

calculator, permițând astfel profesorului să se preocupe cu preponderență de coordonarea activităților didactice, de adaptarea programelor de instruire la specificul disciplinei şcolare şi la profilul avut în vedere, precum şi de aspectele educative ale instruirii. Fără a fi înlocuit de calculator, profesorul îşi va reorienta şi adapta efortul în direcția conducerii procesului de învățământ în condiții specifice noi.

Majoritatea specialiştilor consideră că nu trebuie să ne întrebăm dacă instruirea se îmbunătățeşte prin utilizarea calculatoarelor, ci cum pot fi utilizate mai bine calitățile unice ale acestora, care le deosebesc de alte medii: interactivitatea, precizia operațiilor efectuate, capacitatea de a oferi reprezentări multiple şi dinamice ale fenomenelor şi, mai ales, faptul că pot interacționa consistent şi diferențiat cu fiecare elev în parte [91,106].

Există şi o limitare principială a posibilităților calculatorului în instruirea asistată de calculator, ce nu poate fi depăşită decât prin menținerea unui rol important al profesorului în achiziția, prelucrarea şi utilizarea informațiilor, în acțiunea de formare a deprinderilor şi abilităților practice (de laborator, în rezolvarea problemelor). Se recunoaşte existența limitelor inteligenței artificiale, datorate caracterului complex al limbajului natural, cu încărcătură metaforică şi contextuală (care la comunicarea orală este însoțită şi de gestică, ton etc.) şi care nu se poate prelua integral de către calculator.

Programul de instruire reprezintă nucleul instruirii asistate de calculator. Termenul „program de instruire” se datorează teoriei instruirii programate. Specialiştii prezintă mai multe interpretări a acestei noțiuni. Prin această noțiune se subînțelege „materia de studiu, în care se descriu cunoştințele, deprinderile şi priceperile, care trebuie însuşite, împreună cu metodele lor de însuşire. Ele propun rezolvarea unui volum oarecare de probleme la tema dată sau noțiunea dată, studiate preventiv la prelegere”. După părerea altor autori prin programele de instruire computaționale înțelegem „un astfel de mijloc de instruire, care este purtătorul unui conținut tematic prestabilit sau prezintă obiectele de studiu şi, organizând activitatea atât a profesorilor, cât şi a elevilor, poate fi aplicat în procesul de învățământ doar prin intermediul calculatorului”.

2.5 Clasificarea programelor educaționale după funcția pedagogică

2.5.1 Programe de exersare

Programele de exersare nu sunt concepute pentru a preda noi cunoştințe, ci intervin ca supliment al lecției din clasă. Ele sunt destinate consolidării unui număr limitat de deprinderi specifice unei discipline şcolare [1], prin seturi de sarcini repetitive, urmate întotdeauna de aprecierea răspunsului dat de elev. Există două modalități de realizare informatică a lor:

- exercițiile care vor fi prezentate elevului sunt stocate în memoria calculatorului, de unde sunt extrase într-o ordine predeterminată sau în mod aleatoriu;
- exercițiile nu există ca atare în memoria calculatorului, ci sunt generate de acesta, în conformitate cu un anumit algoritm.

În prima variantă de realizare este absolut necesar de creat o bază destul de vastă de probleme, ele, la rândul lor, fiind clasificate după nivelul de complexitate. Pentru a limita sau chiar a exclude copierea este necesar de introdus în aceste programe un generator de variante de lucrări, care ar lucra în mod aleator. Un atare program are un avantaj destul de serios în comparație cu programul realizat în a doua variantă: profesorul, şi prin urmare calculatorul, cunoaşte răspunsul la problemă, ceea ce facilitează verificarea lucrului îndeplinit de elev.

Varianta a doua are ca avantaj volumul mai mic de memorie, pe care îl ocupă şi imposibilitatea de copiere, dar există pericolul, ca generatorul de probleme să genereze o problemă, care în principiu nu poate fi rezolvată.

2.5.2 Programe de prezentare interactivă de noi cunoştinţe

Programe de prezentare interactivă de noi cunoştinţe încearcă să creeze condiţiile pentru o activitate mai apropiată de dialogul profesor-elev, adică a dialogului dintre cel care învaţă şi mediul special construit pentru a-l ajuta în acest scop. Programul încorporează şi poate prezenta materialul de învăţat pe baza unui anumit tip de interacţiune [18,74]. Dacă această interacţiune este controlată de calculator, atunci se vorbeşte despre un dialog tutorial, iar dacă este controlată de către elev - despre un dialog de investigare. Termenul de tutor este utilizat pentru a desemna tipurile de învăţare, în care activitatea elevului este controlată de calculator. În general un astfel de program funcţionează astfel:

- prezintă una sau mai multe secvenţe de informaţii;
- solicită elevului să răspundă la o întrebare, să rezolve un exerciţiu;
- prezintă aprecierea răspunsului şi introduce secvenţa următoare, ţinând sau nu cont de răspunsul elevului.

Pentru ca prezentarea secvenţelor să nu devină plictisitoare şi pentru a sprijini elevul în organizarea activităţii sale de învăţare, materialul este împărţit în mai multe module, fiecare necesitând pentru parcurgere un interval de 15-20 de minute. De asemenea, se oferă acces la diverse informaţii necesare îndeplinirii sarcinilor de lucru propuse elevului.

Un dialog tutorial bine conceput presupune o aranjare adecvată şi atrăgătoare a materialului, de natură să faciliteze înţelegerea, o bună tehnică în elaborarea întrebărilor sau exerciţiilor, ca şi posibilitatea de a aprecia corect răspunsurile date de elev.

O formă evoluată a interacţiunii didactice este oferită de programele de investigare (de interogare). Dacă un tutor prezintă materialul de învăţat într-o ordine predeterminată, programul de investigare pune la dispoziţia elevului un mediu de unde elevul poate să extragă informaţiile necesare pentru rezolvarea sarcinii propuse, pe baza unui set de reguli. Drumul parcurs de către elev este determinat într-o mare măsură de iniţiativa elevului care învaţă. O investigaţie poate fi orientată spre atingerea unor scopuri precise sau poate fi o explorare.

În practică, cele două moduri nu sunt delimitate strict, ambele putând fi utilizate în cadrul aceluiaşi program. În momentul de faţă, există preocupări de a realiza medii de învăţare care să permită derularea unui dialog natural între cel care învaţă şi calculator; acestea au la bază o abordare bazată pe inteligenta artificială şi sunt cunoscute sub denumirea de instruire inteligentă asistată de calculator. Din acest motiv, astfel de tipuri de programe mai poartă denumirea de manuale electronice. Există unele cerinţe prestabilite de practica elaborării manualelor electronice, pe care obligatoriu trebuie să le satisfacă acest tip de programe de instruire:

- nu este o simplă trecere a unui manuscris sau monografii oarecare pe purtători de informaţie moderni cu vizualizarea ulterioară pe monitorul video. Principala componentă a lui este dialogul activ al instruitului cu calculatorul pe niveluri de complexitate diferite;
- în forma contemporană manualul electronic unifică manualul, culegerea de probleme, ghidul, indicaţiile metodice pentru organizarea şedinţei de instruire, mijloace de evaluare însoţite de criteriile de apreciere a cunoştinţelor;
- este o posibilitate de a diferenţia, individualiza învăţarea şi de a introduce în acest proces specialiştii de forţă în domeniu;
- este baza dezvoltării instruirii interdisciplinare şi a cercetărilor;
- este un complex de mijloace de automatizare a învăţării şi a evaluării cunoştinţelor, de simulare a experimentelor cu o bază informaţională voluminoasă;
- este un sistem ce se autoperfecţionează din contul includerii noilor blocuri şi subsisteme.

2.5.3 Programe de simulare

Programele de simulare permit reprezentarea controlată a unui fenomen sau sistem real, prin intermediul unui model care are un comportament analog. Aceste programe permit variația unor parametri şi observarea modului în care se schimbă comportamentul sistemului ca răspuns la modificările efectuate. Ele permit aducerea în clasă a unor fenomene care nu pot fi prezentate altfel sau a căror realizare experimentală necesită un cost ridicat. În aceste cazuri sunt urmărite mai multe scopuri [1,30]:

- de a determina vitalitatea procesului studiat în diferite situații;
- de a optimiza acest proces;
- de a prognoza evoluția ulterioară a sistemului;
- de a elabora proiectul unei noi instalații, agregat sau sisteme;
- de a efectua procesul de instruire pe bază de modele.

Reprezintă un mijloc, care este foarte ilustrativ şi, comparativ cu costul realizării situațiilor reale în cazul când acest lucru este posibil, este puțin costisitor. Una din variantele posibile ale unui astfel de program îl reprezintă programele pentru realizarea lucrărilor de laborator. Evident, astfel de *software*-uri se utilizează preponderent în discipline ca fizica şi chimia şi sunt aplicate în una din următoarele situații:

- lucrarea de laborator nu poate fi realmente efectuată;
- lucrarea de laborator computerizată completează lucrarea de laborator tradițională.

În ambele cazuri, utilizarea lor este foarte utilă. În primul caz astfel de lucrări computerizate de laborator permit formarea cunoştințelor aplicative (a deprinderilor) în domeniile în care nu este posibilă efectuarea unor lucrări de laborator pe obiecte, utilaje, agregate, procese reale. Astfel, se realizează un model matematic al acestui proces, utilaj, obiect, luând neapărat în considerație toate proprietățile caracteristice lui. Programul se realizează astfel, ca să asigure lucrul elevului sau studentului în regim interactiv, să conțină diferite accesorii (baze de date, calculator, regim grafic interactiv etc.) şi să fie comod şi simplu în utilizare.

În al doilea caz, programele computerizate au menirea de a completa lucrările de laborator tradiționale. În cadrul lecției, aplicând utilajul respectiv, instruitul poate realiza doar o variantă, în cel mai bun caz câteva, a lucrării respective. Dar, pentru a forma nişte cunoştințe şi deprinderi temeinice şi, mai ales pentru a facilita transferul cunoştințelor, este necesară studierea proceselor în diferite situații de realizare, uneori chiar şi în condiții extremale. În cazul acesta, o lucrare computerizată este o ieşire foarte binevenită din situație, reacționând foarte repede la schimbarea parametrilor lucrării introduşi în modelul programat instalației, utilajului sau procesului. Darea de seamă pentru o astfel de lucrare, realizată în două module: experimental şi model matematic, se face integral.

2.5.4 Programe pentru testarea cunoştințelor

Programele pentru testarea cunoştințelor asigură intervenția calculatorului în una sau mai multe dintre etapele verificării cunoştințelor şi reprezintă programatura educațională cu cea mai îndelungată istorie. Scopul principal al elaborării lor a fost eliberarea profesorului de un aşa lucru de rutină, care necesită mult timp şi eforturi cum este evaluarea cunoştințelor [3,43]. Prima încercare a fost realizarea evaluării prin mijloacele instruirii programate, dar a eşuat din cauza mijloacelor tehnice primitive care se utilizau.

Evaluarea este una din componentele de bază ale procesului de instruire şi constă, în esență, în verificarea realizării obiectivelor acestui proces. În funcție de timpul de promovare evaluarea poate fi clasificată în trei tipuri. Evaluarea inițială tradițional se realizează printr-un test sau o lucrare de control la început de curs, cea formativă / curentă – în timpul lecțiilor prin

diferite mijloace şi tehnici de evaluare, iar cea finală – iarăşi printr-un test sumativ la sfârşitul secvenţei de învăţare sau examen la finele unui curs sau la fine de an de studii.

Indiferent de forma promovării şi tipul evaluării, acest lucru necesită din partea profesorului o muncă enormă şi timp, astfel de multe ori nu este posibil de a corecta situaţia, limitându-se doar la o constatare a rezultatelor evaluării [19,24].

În plus, în realizarea oricărui tip de evaluare o importanţă primordială o are alegerea instrumentului de evaluare, care trebuie să posede următoarele calităţi:

- validitate: măsoară ceea ce este destinat să măsoare;
- fidelitate: rezultate aproape constante în cursul aplicării lui succesive;
- aplicabilitate: se administrează şi se interpretează cu uşurinţă;
- obiectivitate: gradul de concordanţă cu rezultatele evaluării efectuate prin alte mijloace sau de alţi evaluatori este maxim.

În condiţiile când acest instrument este realizat de profesor personal, iar rezultatele sunt verificate de asemenea de el, nu se poate da o apreciere reală calităţilor acestui instrument. În multe cazuri asupra profesorului şi, prin urmare, a rezultatelor evaluării influenţează şi factorii subiectivi: emoţiile, oboseală, indulgenţa sau dimpotrivă severitatea etc.

Spre deosebire de profesor, calculatorul electronic nu are nervi, nu are emoţii, el este lipsit de sentimente şi, deci, va realiza procesul de evaluare la un nivel maxim de obiectivitate în cazul utilizării unui program calitativ de evaluare a cunoştinţelor. Astfel un program de evaluare a cunoştinţelor trebuie să asigure: evaluare obiectivă, evaluare diferenţiată şi evaluare individualizată.

Evaluarea formativă sau curentă se realizează, de obicei, în programele de instruire de celelalte tipuri, fie printr-o probă mică de control, fie prin aprecierea interactivă a lucrului instruitului cu programul în cadrul lecţiei curente. Din acest motiv, putem vorbi doar despre programe, care realizează evaluarea iniţială sau cea sumativă.

Reieşind din aceste cerinţe, putem stabili minimumul pe care trebuie să-l conţină un atare program:

- bază impunătoare de probe de evaluare;
- criterii de evaluare;
- diferite niveluri de complexitate cu posibilitatea alegerii lor;
- generator de probleme;
- calculator încorporat;
- posibilitatea introducerii răspunsurilor proprii;
- posibilitatea alegerii tempoului individual de lucru;
- posibilitatea întreruperii în orice moment a procesului de evaluare;
- posibilitatea de formare a unui fişier individual cu rezultatele evaluării, analiza rezultatelor şi recomandările ulterioare.

Astfel, rolul profesorului în procesul de evaluare se reduce doar la supravegherea derulării corecte a procesului de evaluare, a acordării de consultaţii, dacă este necesar. Cel mai eficient lucrează programele adaptative de evaluare a cunoştinţelor, care propun elevilor posibilitatea alegerii nivelului de complexitate dorit, evident cu aprecierea corespunzătoare.

2.5.5 Programe pentru dezvoltarea unor capacităţi şi aptitudini printr-o activitate de joc

Jocurile computerizate sunt o parte indisolubilă a programaturii unui calculator. Fiecare joc computerizat dezvoltă unele calităţi ale personalităţii şi există o clasificare preponderent simbolică a jocurilor, printre care putem menţiona: jocurile logice (şah, dame, madjong, tetris), jocuri strategii, jocuri de acţiune etc.

Jocuri didactice sau programatura pentru dezvoltarea unor activităţi şi aptitudini printr-o activitate de joc sunt destul de puţine relativ la numărul total de jocuri, iar printre ele şi mai puţine sunt de calitate corespunzătoare. Aceste programe au menirea de antrenare a cunoştinţelor şi deprinderilor de a le utiliza, aducându-le pe ultimele până la automatism. Este aproape imposibil de a realiza acest lucru printr-o programă de instruire de alt tip sau printr-un alt mijloc de instruire, deoarece se presupune efectuarea repetată a unor acţiuni, care implică monotonia procesului de instruire şi pierderea de interes a instruitului.

Oricare program de acest tip presupune existenţa unui scenariu şi a unor reguli de joc. El trebuie să asigure:

- satisfacţie în rezultatul jocului;
- competitivitate;
- atractivitate;
- aprecierea performanţelor atinse;
- mai multe grade de libertate a jucătorului.

Jocurile computaţionale didactice pot fi clasificate în două grupe mari [1,22]:

- jocuri de înrolare – presupune încadrarea jucătorului cu un anume rol într-o situaţie, care simulează o situaţie reală (în economie, administrare publică sau militărie), unde el se manifestă ca specialist în acest domeniu.
- jocuri logice – sunt de regulă jocuri de tipul jocului de şah, altfel spus, există un câmp de joc, care conţine „figuri”, care se cer aranjate într-o anumită ordine sau selectate într-o anumită componenţă.

Câteva din problemele principale la elaborarea unor astfel de programe sunt: complexitatea elaborării lor, stabilirea unei variante inedite de joc, sfera limitată de deprinderi, pe care ele le dezvoltă şi pericolul ca elevul să se pasioneze de joc în detrimentul celorlalte preocupări . Astfel de programe de obicei se propun pentru utilizare independentă acasă sau în cadrul unei lecţii de recapitulare în clasele mici.

În afară de programele educaţionale menţionate, care pot fi utilizate în întregime în cadrul procesului de studiu a unor discipline concrete, în învăţământ se mai aplică şi programele pentru gestionarea procesului instructiv-educativ. Aplicarea acestor sisteme computerizate de automatizare ar face managementul instituţiei date mai operativ şi mai calitativ.

În concluzie, se poate afirma că instruirea asistată de calculator conferă învăţământului tradiţional o nouă dimensiune, ce promite creşterea substanţială a randamentului învăţării şcolare, oferind astfel o soluţie realizabilă şi realistă problemelor cu care se confruntă învăţământul contemporan [60].

Din studiile întreprinse pe plan internaţional s-au desprins o serie de concluzii interesante cu privire la eficienţa utilizării programaturii educaţionale, dintre care amintim:

- aproape toate cercetările relevă avantajele utilizării calculatoarelor în comparaţie cu alte metode;
- reducerea timpului de studiu;
- atitudinea fata de calculator se modifică pozitiv;
- utilizarea calculatoarelor este mai eficientă în ştiinţe decât în domeniul limbilor;
- în instruirea asistată de calculator, exersarea este eficientă în formarea deprinderilor elementare, în timp ce sistemele tutoriale sunt mai eficiente în formarea deprinderilor intelectuale de nivel superior;
- instruirea asistata de calculator este mai eficientă ca instruire complementară, decât ca formă alternativă;
- elevii care învaţă lent şi cei rămaşi în urmă câştigă mai mult decât cei fruntaşi;
- strategiile bazate pe utilizarea calculatoarelor sunt mai eficiente la nivelurile inferioare.

2.6 Calculatorul la clasa de matematică

Calculatorul poate fi o unealtă naturală pentru matematicieni. La întrebarea : „trebuie să fie utilizat calculatorul la clasa de matematică?”, rezultatul ar fi o divizare în două categorii. Unii văd în utilizarea calculatorului o mare oportunitate, pretinzând că permite exploatarea unor teme matematice care erau imposibil de abordat înainte la clasă. Alții văd un pericol că elevii şi studenții vor deveni analfabeți matematic, că vor trata aceste unelte *software* precum cutii negre perfect implementate şi vor sfârşi prin a nu înțelege matematica pe care aceste cutii negre o implementează.

Care față a acestei dezbateri este corectă? Precum toate întrebările sociale dificile, răspunsul este ambele şi niciuna. Punctul de vedere optimist ne lasă să vedem că se pot obține multe câştiguri, iar punctul de vedere pesimist ne arată defectele care trebuie evitate. Trebuie găsită o poziție între extreme unde se poate obține cât mai mult bine şi cât mai mult rău [17].

Una din întrebările frecvente care vin adesea în discuție este relativ la cutiile negre, dacă sunt bune, dacă nu putem avea cutii negre sau studenții nu vor să învețe. Prin cutie se înțelege un proces care reacționează la intrări de date prin producerea unor anumite rezultate sau printr-o comportare de un anumit tip. Cutia este neagră când nu putem vedea ce se întâmplă în interior. Sperăm că se va petrece totul corect, dar nu ştim cu siguranță. Un *software* matematic este adesea privit ca o cutie neagră care rezolvă probleme matematice. Acest fapt poate conduce la pierderea capacității de înțelegere a noțiunilor matematice. De aceea *software*-ul matematic trebuie să prevadă posibilitatea de explorare a fiecărei cutii negre de către utilizatori.

În matematică dacă avem o cutie neagră, atunci anumite concepte matematice sunt închise în ea. Dacă elevii sau studenții nu sesizează modul de lucru intern al cutiei, atunci ei nu pot învăța acele concepte intrinseci implementării cutiei negre. Se poate lucra eficient cu aceste cutii negre la diferite teme, dar dacă învățarea matematicii este obiectivul real, atunci performanța şi eficiența sunt scopuri minore. Pe de altă parte, dacă se dă elevilor şi studenților un volum mare de probleme relativ simple adecvate pentru manipulare algebrică manuală, ei vor deprinde o îndemânare prin repetiție, dar nu întotdeauna au timpul necesar pentru teme matematice mai interesante.

Să considerăm o materie la care studenții învață despre diferențiere, limite, integrări şi serii. Prin aceasta studenții iau contact cu analiza reală şi complexă, topologie, logică formală. Aceste topici mai generale nu sunt explicit abordate, dar sunt în spatele conceptelor matematice care sunt prezentate. Cu alte cuvinte, se tratează matematica mai formală şi riguroasă precum cutii negre, dacă nu explicit, atunci prin nemenționarea presupunerilor. Cutiile negre pot fi folosite astfel pentru a ascunde complexitatea de studenți când aceştia nu sunt pregătiți. Aceasta permite profesorilor să predea concepte înalte, precum subrutinele cutii negre permit programatorilor să scrie cod de nivel înalt. De aceea cutiile negre nu pot fi privite ca un lucru rău, ele sunt un fapt necesar în viața educațională. Procesul de construcție a unui curs poate fi văzut ca o decizie care cutii negre să fie prezentate şi cărora li se va ignora existența.

Însă cutiile negre pot deveni lucruri rele în următoarele situații:

- când studenților li se interzice accesul în interiorul cutiilor negre (contra-exemplu: în Maple se poate vizualiza, pas cu pas integrarea unei funcții, iar studenții pot interveni în această prezentare);
- când studenții nu înțeleg cum o cutie neagră lucrează chiar dacă este deschisă (acesta este cel mai important fapt pe care trebuie să-l ia în considerație profesorii; ei trebuie să cunoască care cutii negre trebuie deschise, contextul utilizării şi relevanța lor);
- când studenții nu au motive pentru a le păsa de interiorul unei cutii ce conține concepte critice pentru subiect.

Cum arată o clasă de matematică ideală? Astăzi adesea este un loc în care studenţii învaţă matematică într-o manieră pas cu pas, memorează proceduri şi lucrează în izolare faţă de ceilalţi studenţi [40, 80]. Trebuie să fie însă o comunitate matematică unde studenţii şi profesorul discută în limbaj matematic, lucrează în grupuri pentru a rezolva probleme din lumea reală, chiar probleme deschise, utilizează calculatoarele în mod inteligent, verifică răspunsurile împreună utilizând raţionarea matematică. Testele de aptitudini practice sunt însă o frână în modalitatea de transformare în acest ideal al clasei, influenţând modalitatea de predare. Copiii angajaţi în probleme reale sunt mult mai motivaţi în a învăţa şi a utiliza deprinderi matematice. Performanţele profesorilor trebuie să se îndrepte de la modelul autoritar transmitere de cunoştiinţe la metode centrate pe elev cu stimularea învăţării şi explorarea activă.

Calculatorul este perfect capabil să creeze imagini vizuale ceea ce conduce la îmbunătăţirea gândirii spaţiale necesare în matematică. Majoritatea *software*-urilor educaţionale de pe piaţă sunt concepute să suplimenteze programa matematică existentă. Mediile de rezolvare a problemelor tind să se concentreze asupra problemelor. Înlocuirea treptată a hârtiei şi creionului cu calculatorul este o consecinţă a progresului.

Calculatoarele elimină calculele lungi de rutină şi permit o implicare mai mare în procesul de învăţare. Educaţia matematică trebuie să formeze studenţi care să dea o valoare matematicii, să devină încrezători în abilităţile de a face matematică, să devină rezolvitori de probleme matematice, să comunice matematic, să înveţe să raţioneze matematic. Calculatorul poate avea un rol important în realizarea acestor deziderate.

2.7 Considerente psiho-pedagogice ale utilizării calculatorului în procesul de studiere a matematicii

În procesul de învăţământ pot fi evidenţiate trei elemente de bază, care interacţionează: elevul / elevii, profesorul şi conţinutul disciplinei [25,33], aceste elemente fiind unite într-o schemă liniară. Cunoştinţele sunt deţinute de o singură persoană: profesorul şi ele pot fi transmise elevului numai prin intermediul profesorului. Deoarece elevul nu are acces direct la conţinuturi, nu este obligatoriu ca ele să fie prezentate sub o formă comprehensibilă.

Într-o abordare mai contemporană, elementele pot fi unite într-o schemă triunghiulară. În această schemă, relaţia între profesor şi elev este obiectivizată prin acţiune de formare, relaţia dintre profesor şi conţinuturi este obiectivizată prin acţiunea de predare, iar relaţia dintre conţinuturi şi elev este obiectivizată prin acţiunea de învăţare. În schema triunghiulară sunt posibile diferite moduri de acces ale elevilor la conţinuturi.

Prima variantă coincide cu schema liniară. Modelul pedagogic respectiv se numeşte model transmisiv. Pentru a face posibilă achiziţionarea conţinutului de către elev, profesorul trebuie să realizeze două activităţi:

- adaptarea conţinutului predat (transformarea conţinutului ştiinţific într-un conţinut de învăţat); activitatea respectivă poartă denumirea de transpoziţie didactică;
- organizarea conţinutului într-o suită de mesaje transmise elevilor.

În modelul transmisiv profesorul este depozitarul exclusiv al informaţiei necesare elevului. Prin urmare, accesul elevului la conţinut depinde nu numai de prezenţa profesorului, dar şi de stilul său pedagogic, de concepţia învăţării adoptate de el etc. Fiind antrenat în procesul de transmitere a informaţiei, profesorul nu reuşeşte să se implice în procesul de dirijare a activităţii de învăţare a elevilor (evidenţierea celor ce întâmpină dificultăţi cognitive, înţelegerea naturii acestor dificultăţi şi acordarea ajutorului necesar).

Varianta a doua constă în crearea condiţiilor de acces direct şi autonom ale elevului la conţinutul studiat. Modelul pedagogic respectiv se numeşte modelul însuşirii active. În acest caz conţinutul disciplinei trebuie expus într-o formă comprehensibilă, fie pe suport de hârtie, fie pe suport numeric. Punerea conţinutului disciplinei într-o formă publicitară comprehensibilă se numeşte mediatizare. Mediatizarea este o condiţie necesară pentru modelul însuşirii active. Fiind eliberat de funcţia de transmitere a informaţiei, profesorul îşi poate concentra atenţia asupra procesului de învăţare a elevului – activitatea de mediere. Este evident că activitatea de mediere este imposibilă fără activitatea de mediatizare.

Modelul însuşirii active presupune că elevii dispun de abilităţi de învăţare autonomă. Dacă aceste abilităţi nu sunt încă formate, atunci este posibilă o variantă mixtă a organizării procesului de învăţământ prin îmbinarea modelului transmisiv şi a însuşirii active. O asemenea variantă mixtă va fi analizată în continuare.

Analiza literaturii psiho-pedagogice referitoare la problema utilizării calculatorului în procesul de studiere a matematicii arată că acest aspect al instruirii are o importanţă deosebită [68,100]. O parte din elevi termină studierea matematicii în şcoala medie sau în liceu, din acest motiv este necesar de creat o rezervă de cunoştinţe matematice suficientă pentru activitatea ulterioară a absolventului. Mai mult ca atât, utilizarea calculatorului în procesul de studiere a matematicii realizează şi nişte obiective de ordin general, este o metodă eficientă de a pregăti specialişti sau absolvenţi capabili:

- să se adapteze rapid situaţiilor vieţii, să obţină independent cunoştinţele necesare, să le folosească în practică pentru rezolvarea diferitelor probleme, pentru a se integra fără probleme în medii profesionale şi sociale;
- să judece independent, critic, să poată vedea problemele apărute şi să găsească căi raţionale de rezolvare a lor; să conştientizeze, unde şi în ce fel cunoştinţele căpătate pot fi aplicate; să fie capabil să genereze idei noi, să gândească creativ;
- să ştie a lucra cu informaţia (să poată acumula faptele necesare pentru rezolvarea problemei concrete, să le analizeze, să elaboreze ipoteze de rezolvare, să realizeze generalizările necesare, să efectueze concordanţa dintre metodele analogice şi alternative de rezolvare, să stabilească legităţile statistice, să facă concluzii argumentate, să folosească concluziile obţinute pentru rezolvarea problemelor noi);
- să fie comunicabil şi să se poată încadra în diferite grupe sociale fără dificultăţi.

În varianta triunghiulară rolul principal îi revine elevului, care este obiectul instruirii, dar şi subiectul propriei învăţări. Cunoaşterea personalităţii elevului devine o condiţie necesară pentru proiectarea şi realizarea unui proces de instruire eficient. Luarea în consideraţie a acestor particularităţi impune individualizarea procesului de învăţământ, alegerea metodelor optime de lucru şi deplasarea accentelor în cursul studiat pentru fiecare caz.

Din punct de vedere psiho-pedagogic principala problemă, care apare la utilizarea tehnologiilor informaţionale în procesul de instruire constă în revizuirea aparatului conceptual de descriere a diferitor niveluri de reflectare a informaţiei. Astfel, există părerea unei anumite părţi a specialiştilor că utilizarea instruirii asistate de calculator schimbă esenţial sensul verbului „a cunoaşte”. Noţiunea „de a memora informaţia” se transformă în „de a primi acces la informaţie”. Structura gândirii la un elev instruit tradiţional este condiţionată de structura textului tipărit, care are următoarele particularităţi: liniaritate, analiticitate, raţionalitate. În mediul de imitaţie, creat de calculator se stimulează imaginaţia, flexibilitatea, conexiunea.

Cercetătorul Gurieva evidenţiază consecinţele psihologice ale computerizării activităţii umane: funcţionale, ontogenetice şi istorice [26]. Consecinţele funcţionale constau în următoarele: în rezultatul transferării unor funcţii umane spre realizare calculatorului şi a păstrării de către om a deprinderilor, cunoştinţelor şi atitudinilor tradiţionale la individ se poate forma o predispoziţie spre utilizarea calculatorului sau dimpotrivă – o barieră psihologică în

această direcție. În acest sens pot fi evidențiate câteva tipuri de atitudini privitor la utilizarea calculatorului:

- indiferența (această tehnică pe mine nu mă privește, eu n-o cunosc și nici nu trebuie s-o cunosc). Această atitudine a existat atunci când tehnica de calcul era utilizată de un grup mic de specialiști. Perioada manifestării ei începe în anii '40 și a durat până în anii '60 a secolului trecut.
- negativismul și chiar agresivitatea, manifestată în special de populația adultă (mie nu îmi trebuie, tot ce poate face calculatorul este numai ceva ideal, în realitate însă el nu poate face așa ceva). Această atitudine apare atunci când calculatorul a început să fie utilizat pe larg în activitatea profesională. Se simțea inerția de modificare a mentalității, când persoana trebuia să se dezică de unele deprinderi deja formate, să și le modifice sau chiar să însușească ceva nou. Această perioadă a durat între anii '60 - '80, finalizând cu apariția calculatoarelor personale;
- pozitivismul și necesitatea (calculatorul a devenit o parte indispensabilă a vieții cotidiene, pentru unii devenind domeniu permanent de activitate, iar pentru alții - un mijloc de realizare periodică a anumitor activități necesare). In această perioadă, care durează începând cu anii '80 și până în zilele noastre, a dispărut teama de a lucra la calculator; dimpotrivă, majoritatea populației manifestă dorința de a lucra cât mai mult la el. Au apărut cunoștințe sigure despre posibilitățile calculatorului, el fiind utilizat tot mai eficient [4].

Consecințele ontogenetice ale computerizării se analizează în perspectiva dezvoltării personalității utilizatorului calculatorului. Studierea procesului de formare a calităților unei personalități sub influența calculatorului arată că în afară de factorii obiectivi (tipul și volumul problemei care se rezolvă, periodicitatea și forma de utilizare a calculatorului, existența sau lipsa dificultăților în procesul de utilizare a programelor) asupra dezvoltării ontogenetice a personalității influențează și particularitățile calităților personalității transmise genetic.

Consecințele istorice sunt studiate sub aspectul influenței computerizării asupra dezvoltării omului din punct de vedere psihologic. Calculatorul este prezentat ca o unealtă nouă, care servește ca intermediator al activității umane, căreia i se transmite unele din funcțiile pe care le îndeplinea tradițional creierul uman (calcule, modelări, unele aprecieri etc.) .

Calculatorul ca unealtă de activitate poate conduce atât la dezvoltarea calităților individuale ale elevului, formându-i nu numai niște capacități de creativitate, noi după conținut și particularități dinamice, cât și la formalizarea, șablonarea gândirii lui, dezvoltarea unei personalități fără inițiativă în dependență de calitatea și direcționarea programelor computaționale. Cu alte cuvinte, utilizarea calculatorului în procesul de instruire poate aduce nu numai la consecințe progresiste, ci și regresiste în dezvoltarea psihicului elevului [100].

Pentru utilizarea eficientă a calculatorului în procesul de instruire este necesar de studiat integral complexul de probleme psiho-pedagogice care pot apărea în interacțiunea elev-calculator, în particular. În opinia noastră aceste probleme sunt următoarele:

- influența calculatorului asupra motivației de învățare a elevilor și formarea la ei a tendințelor spre cunoaștere;
- consecințele medicale (nepsihologice) și psihologice ale utilizării tehnologiilor informaționale;
- dezvoltarea personalității în condițiile utilizării metodologiilor computaționale;
- schimbarea rolului profesorului în procesul de instruire;
- formarea unor tipuri noi de relații între elevi, între elevi și profesori;
- influența calculatorului asupra perceperii informației.

Analiza publicațiilor referitoare la această temă arată că utilizarea calculatorului în instruire are atât avantajele, cât și dezavantajele sale. Profesorul trebuie să cunoască avantajele, dar și dezavantajele, pentru a diminua efectul lor nefast. Progresul tehnologiilor computaționale micșorează posibilitatea apariției emoțiilor negative la utilizarea ei, dar nu ridică bariera psihologică, care apare la interacțiunea unor persoane cu tehnica modernă.

2.8 *Software* matematic

Noțiunea de ”*Software* matematic” [43,88] se referă la proiectarea şi utilizarea unor sisteme *software* destinate rezolvării unor probleme ce apar în ştiință şi tehnică. Aceste sisteme sunt concepute astfel încât să asiste utilizatorul în efectuarea unor calcule complicate şi/sau voluminoase corespunzătoare unor probleme reale.

Principala motivație a proiectării sistemelor de *software* matematic o reprezintă necesitatea de a obține rapid şi fără erori rezultate pentru probleme de dimensiuni mari. Prelucrări de rutină, cum este rezolvarea unui sistem liniar, pot fi realizate manual doar dacă dimensiunea problemei (de exemplu, a sistemului) este rezonabilă. Cu ajutorul calculatorului, însă, pot fi rezolvate sisteme care au sute şi mii de ecuații. Pe de altă parte rezolvarea unor probleme necesită utilizarea unor metode matematice ce nu sunt întotdeauna la îndemâna specialiştilor dintr-un domeniu ştiințific sau tehnic oarecare. *Software*-ul matematic elimină această dificultate întrucât are implementate o clasă largă de metode matematice, utilizatorul trebuind doar să le apeleze fără să fie necesar să cunoască detaliile implementării lor.

Nu trebuie neglijat faptul că aceste sisteme pot fi utilizate şi în scop didactic, atât la nivel elementar cât şi la cel universitar, pentru predarea de discipline ştiințifice (matematică, fizică, chimie), în special în cadrul învățământului la distanță.

Module *software* pentru prelucrări matematice au fost realizate încă din anii 1960 când calculatoarele au început să devină un instrument în rezolvarea problemelor tehnico-ştiințifice. Acestea erau însă dedicate unui anumit tip de probleme şi erau grupate în biblioteci de unde erau apelate de către programe scrise în limbaje de uz general. Adesea setul de parametri ai acestor module era mare iar utilizatorul trebuia sa cunoască cel puțin câteva dintre detaliile de realizare a modulului.

Sistemele integrate, care permit abordarea unei palete largi de calcule au apărut în anii 1980 o dată cu dezvoltarea puterii de calcul şi evoluția conceptelor de programare. Oferind adesea o interfață uşor de utilizat, aceste sisteme permit utilizatorului să se concentreze mai mult asupra problemei şi mai puțin asupra metodelor matematice de rezolvare a acesteia.

Prelucrările matematice pot fi grupate în următoarele categorii:

- Numerice: Rezultatele calculelor numerice sunt numere. Exemple de astfel de prelucrări sunt: calculul integralei definite a unei funcții, determinarea rădăcinilor unui polinom cu coeficienți numerici, determinarea limitei unui şir numeric etc.
- Simbolice: Rezultatele calculelor simbolice sunt de regulă expresii algebrice sau chiar propoziții matematice. Exemple de astfel de prelucrări sunt: calculul primitivei unei funcții, determinarea rădăcinilor unui polinom cu coeficienți simbolici, efectuarea unui raționament logic etc.
- Grafice: Rezultatele acestor prelucrări sunt de fapt reprezentări grafice ale unor funcții, curbe, suprafețe sau alte obiecte grafice descrise fie prin ecuații, fie prin punctele pe care le conțin etc.

Sistemele de *software* matematic oferă posibilitatea efectuării fiecăreia dintre aceste prelucrări. Unele prelucrări pot fi efectuate direct existând comenzi specifice, iar altele pot fi descrise în limbajul de programare specific sistemului. Spre deosebire de limbajele de programare de uz general sistemele de *software* matematic conțin un limbaj de comandă mult mai bogat în sensul că pot fi specificate printr-o singură comandă şi prelucrări bazate pe algoritmi relativ complicați (de exemplu: inversarea unei matrice, rezolvarea simbolică sau numerică a unui sistem de ecuații diferențiale etc).

La ora actuală există o multitudine de pachete *software* destinate efectuării de prelucrări numerice, simbolice sau grafice.

Unele au caracter mai general oferind facilități care permit utilizarea lor în diverse domenii, iar altele sunt orientate spre sarcini precise şi oarecum limitate la un anumit domeniu, cum sunt, de exemplu :

- pachetele de reprezentări grafice : GnuPlot ;
- programele de prelucrare a datelor experimentale : TableCurve, Origin, DataFit, GnuFit etc ;
- pachetele destinate prelucrărilor statistice : Statistica, SPSS, Splus ;
- sistemele de rezolvare a problemelor de optimizare : MinOpt ;
- sistemele de rezolvare a ecuațiilor diferențiale.

2.9 Contribuții la realizarea unor programe educaționale destinate studierii unor capitole ale matematicii, fizicii şi informaticii

Înainte de a proiecta sistemul informatic destinat procesului de asimilare a cunoştinţelor din domeniul geometriei, am realizat, în colaborare cu alte cadre didactice, diverse programe educaționale care pot fi utilizate în studiul informaticii, matematicii şi fizicii. Pentru implementarea programelor s-au folosit medii de programare cum ar fi: Delphi 7.0, C++ Builder 6.0, limbajul Java şi programe de simulare precum Matlab şi PSCAD.

În cazul informaticii am implementat programe ce pot fi utilizate în cadrul disciplinelor „Structuri de date şi algoritmi” şi „Limbaje formale”. Programele destinate primei discipline prezintă metoda „Divide et impera” [45], metodele de sortare [84], arborii binari [85] şi listele liniare [86]. Pentru disciplina „Limbaje formale” am realizat un program care prezintă gramaticile stohastice şi aplicațiile acestora în generarea fractalilor [52].

Aplicația informatică care prezintă metoda „Divide et impera” este realizată folosind resursele mediului de programare Borland Delphi. Fereastra principală a aplicației conține un meniu cu următoarele opțiuni: prezentare teoretică, probleme rezolvate prin această metodă şi un test grilă pentru verificarea cunoştinţelor însuşite. Dintre problemele rezolvate prin această metodă sunt prezentate: determinarea valorii minime dintr-un şir de numere prin simulare grafică a stivei utilizate de această metodă (Figura 2.9.1.a), metoda de sortare rapidă a elementelor unui şir de numere şi jocul „Turnurile din Hanoi” (Figura 2.9.1.b).

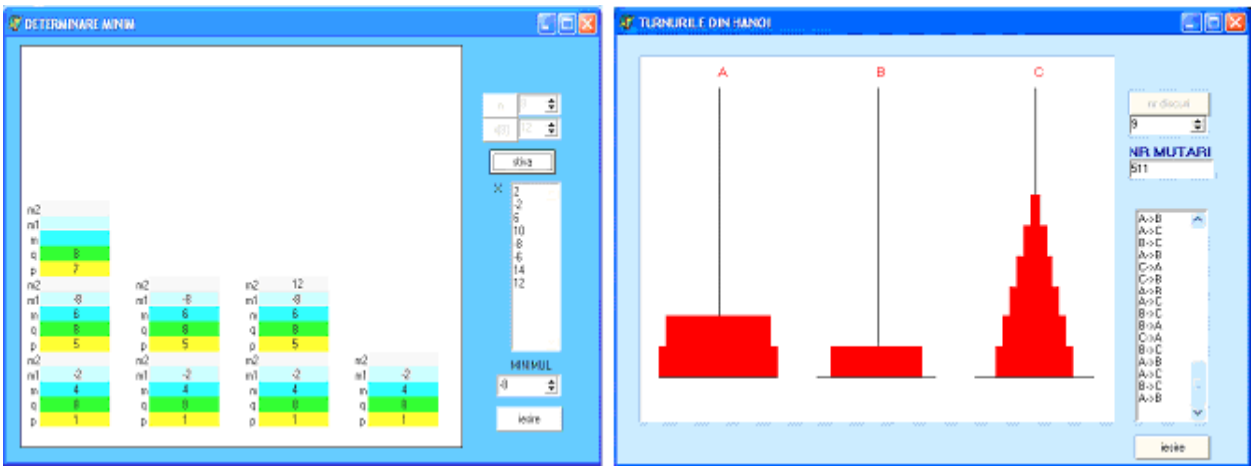


Figura 2.9.1. Determinarea minimului unui şir şi jocul „Turnurile din Hanoi”

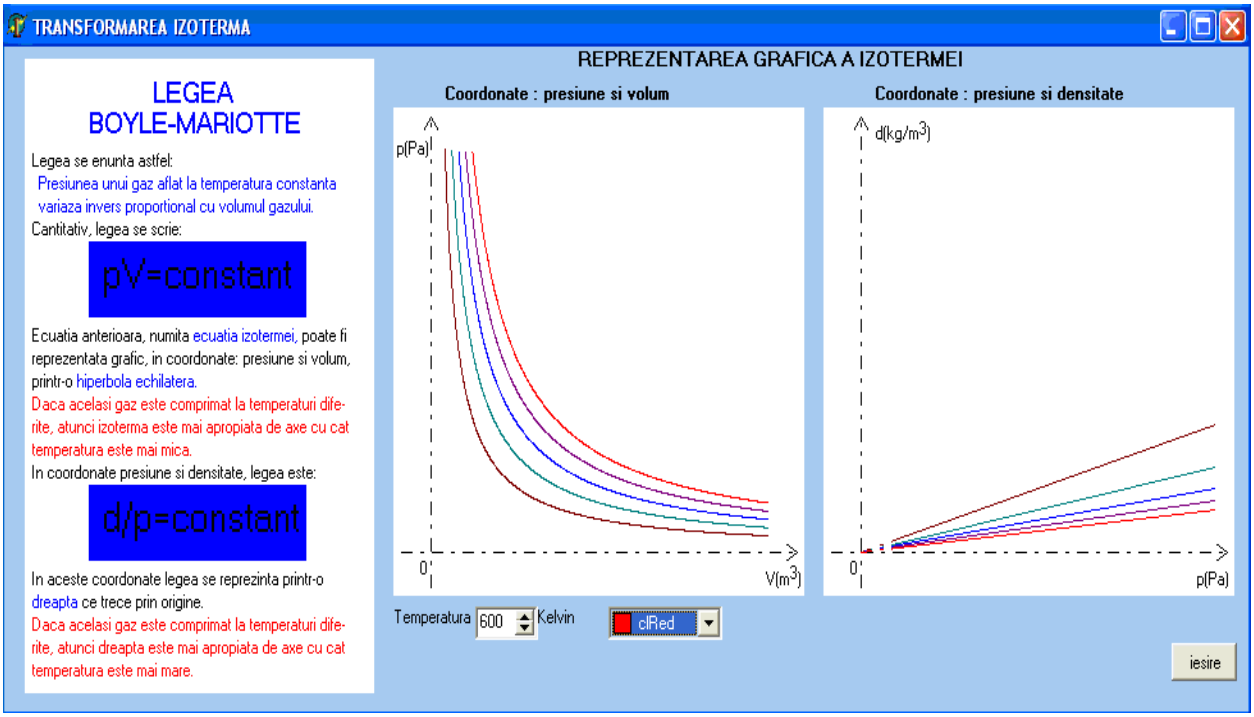


Figura 2.9.2. Transformarea izotermă

Programele destinate studiului fizicii prezintă legile gazului ideal [51], lentile convergente şi divergente [83], puterea electrică [82] şi ciocnirile perfect elastice şi plastice. Aplicaţia informatică care prezintă legile gazului ideal este realizată folosind resursele mediului de programare Borland Delphi. Fereastra principală cuprinde opţiuni corespunzătoare celor trei transformări: izotermă (Figura 2.9.2.), izocoră şi izobară.

Aplicaţia informatică care prezintă ciocnirile perfect elastice şi plastice este realizată folosind resursele mediului de programare Borland C++ Builder 6.0. Fereastra principală cuprinde un meniu care conţine opţiuni pentru prezentare teoretică, aplicaţii pentru cele două tipuri de ciocniri (Figura 2.9.3.) şi test grilă pentru verificarea cunoştinţelor dobândite.

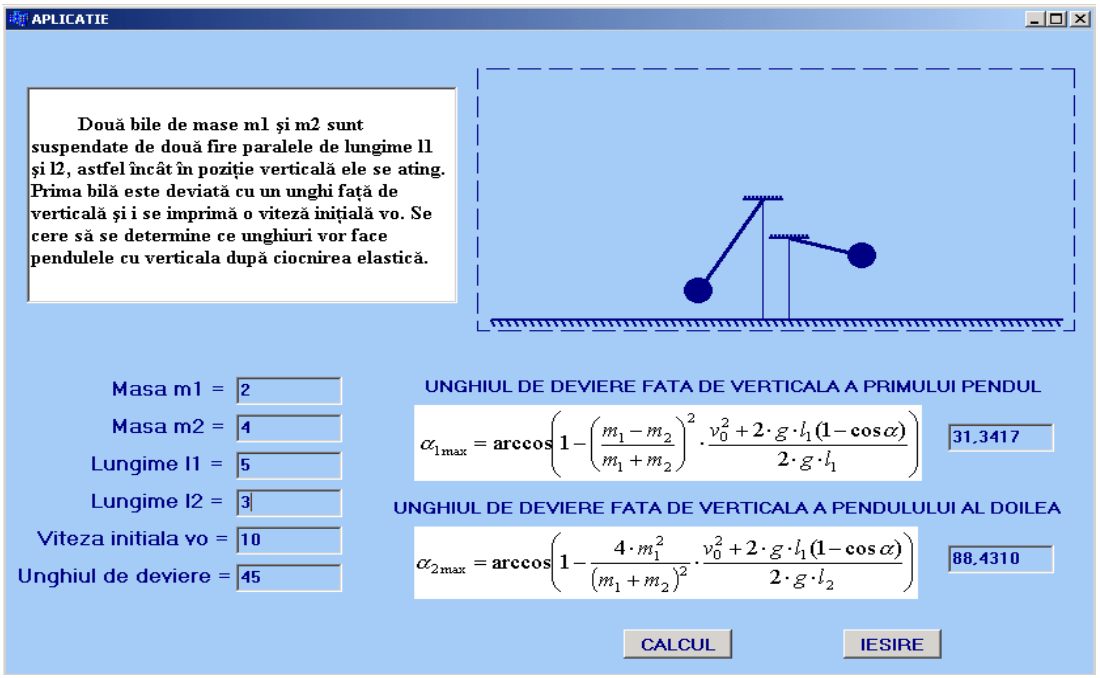


Figura 2.9.3. Aplicaţie pentru ciocnirea perfect elastică

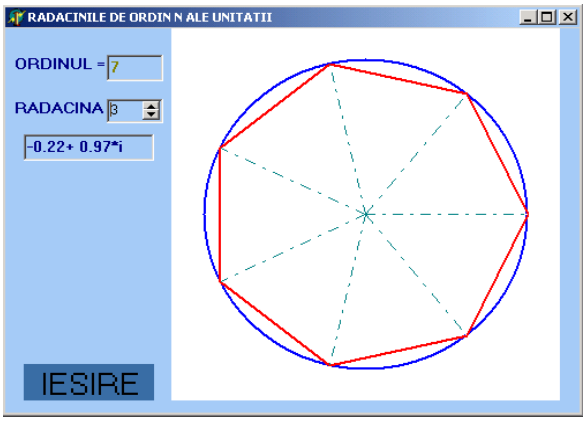


Figura 2.9.4. Rădăcinile de ordin „n” ale unității

Pentru disciplinele „Analiză matematică”, „Algebră”, „Statistică” şi „Geometrie” am implementat programe corespunzătoare următoarelor noțiuni: distribuția binomială [46], numerele complexe [47], grafice de funcții [48], locuri geometrice [49], distribuția normală [50].

Programul care prezintă numerele complexe şi aplicațiile acestora în geometrie este realizat folosind mediul de programare Borland Delphi. Fereastra principală cuprinde opțiuni care prezintă: forma algebrică şi forma trigonometrică a unui număr complex, adunarea şi înmulțirea a două numere complexe, rădăcinile de ordin „n” ale unității (Figura 2.9.4.) şi utilizarea numerelor complexe la rezolvarea unor probleme de geometrie.

2.10 Analiza criticii a unor sisteme de instruire existente pentru geometrie

2.10.1 Manipula Math With Java

Manipula Math with Java, realizat de către International Education Software [44], prezintă noțiuni geometrice care pot fi utilizate în şcolile generale, licee, universitații. Se va observa că este un program interactiv cu multe animații care pot ajuta la înțelegerea noțiunilor geometriei euclidiene. *Applet*-urile programului sunt grupate după cum urmează :

- Geometry 1 - cele 39 de *applet*-uri ce formează acest set cuprind informații legate de următoarele noțiuni (Figura 2.10.1):
  - Drepte paralele şi unghiuri;
  - Figuri congruente şi triunghiuri;
  - Patrulatere şi arii;
  - Figuri asemenea.

- Geometry 2 - cele 42 *applet*-uri ce formează acest set cuprind informații legate de următoarele noțiuni şi relații (Figurile 2.10.2 şi 2.10.3): Cercuri şi teoremele lui Pitagora.

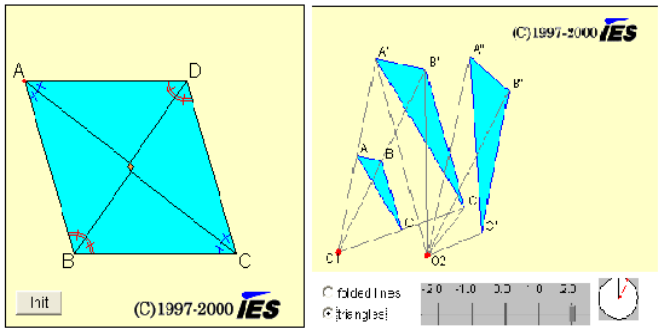


Figura 2.10.1. *Applet*-uri pentru paralelogram şi figuri asemenea în Manipula Math

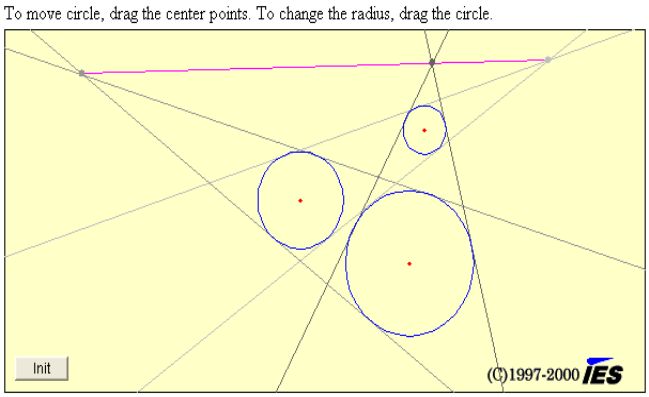


Figura 2.10.2. *Applet* pentru prezentarea unei probleme cu cercuri în Manipula Math

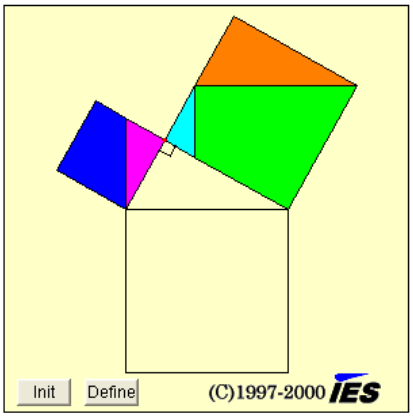


Figura 2.10.3. *Applet* pentru prezentarea teoremei lui Pitagora în Manipula Math

➢ Conics - cele 9 *applet*-uri ce formează acest set cuprind informații legate de următoarele noțiuni (Figura 2.10.4):

- Parabola şi proprietăți ei;
- Elipsa şi proprietățile ei;
- Hiperbola şi proprietățile ei.

Avantaje ale *software*-ului prezentat:

- Materialul teoretic în program corespunde cerințelor programei de studiu, sunt preconizate câteva niveluri de expunere în dependență cu nivelul de pregătire al elevilor;
- Noțiunile sunt prezentate dinamic, prin intermediul *applet*-urilor;
- Teoremele sau rezultatele prezentate sunt însoțite de demonstrație;
- Simplitatea lucrului cu programul.

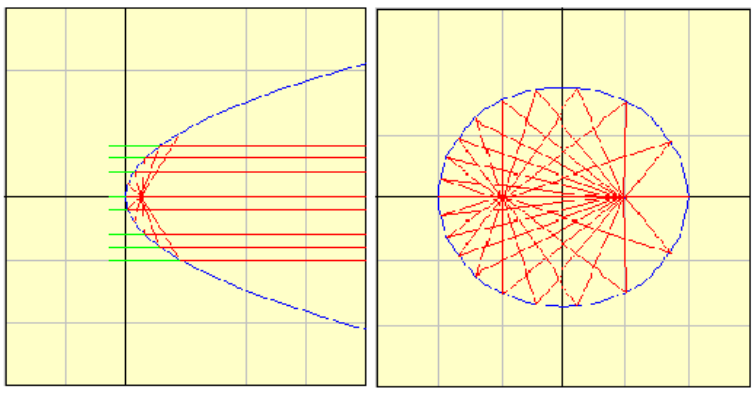


Figura 2.10.4. *Applet* pentru prezentarea proprietăților parabolei şi elipsei în Manipula Math

Dezavantajele *software*-ului prezentat:

- Structurarea materiei nu este reuşită;
- Programul poate fi utilizat numai în etapa de însuşire de noi cunoştinţe; nu cuprinde opţiuni pentru etapele de consolidare a cunoştinţelor dobândite, verificare şi evaluare;
- Nu este oferită posibilitatea de rezolvare interactivă a problemelor;
- Este realizat la un nivel înalt, dar există neajunsuri metodice.

2.10.2 Geometria 8

Geometria 8 este un *software* educaţional realizat de către Vasile Roman şi comercializat de către editura Edusoft [96] şi poate fi utilizat ca material auxiliar la predarea geometriei în spaţiu la clasa a VIII-a. Cerinţele *software* pentru utilizarea programului sunt: sistemul de operare Windows 95, 98, NT sau XP, Internet Explorer şi Flash Player.

Prezentarea informaţiilor este grupată pe cinci capitole:

➢ Capitolul 1. Punct, dreaptă, plan. Axiomele geometriei euclidiene.
În acest capitol se studiază (Figura 2.10.5):

- noţiunile fundamentale ale geometriei euclidiene;
- axiomele geometriei euclidiene;
- relaţii între noţiunile fundamentale ale geometriei euclidiene;
- proprietăţi ale noţiunilor fundamentale din geometria euclidiană.

➢ Capitolul 2. Teoreme de paralelism. Poziţiile relative a trei plane.
În acest capitol se studiază:

- teoreme de paralelism;
- poziţiile relative ale trei plane.

➢ Capitolul 3. Perpendicularitate în spaţiu.
În acest capitol se studiază (Figura 2.10.6):

- definiţiile şi teoremele referitoare la perpendicularitatea în spaţiu;
- teorema celor trei perpendiculare, precum şi reciprocele acesteia.

➢ Capitolul 4. Proiecţii ortogonale.
În acest capitol se studiază:

- proiecţii ortogonale pe o dreaptă şi pe un plan;

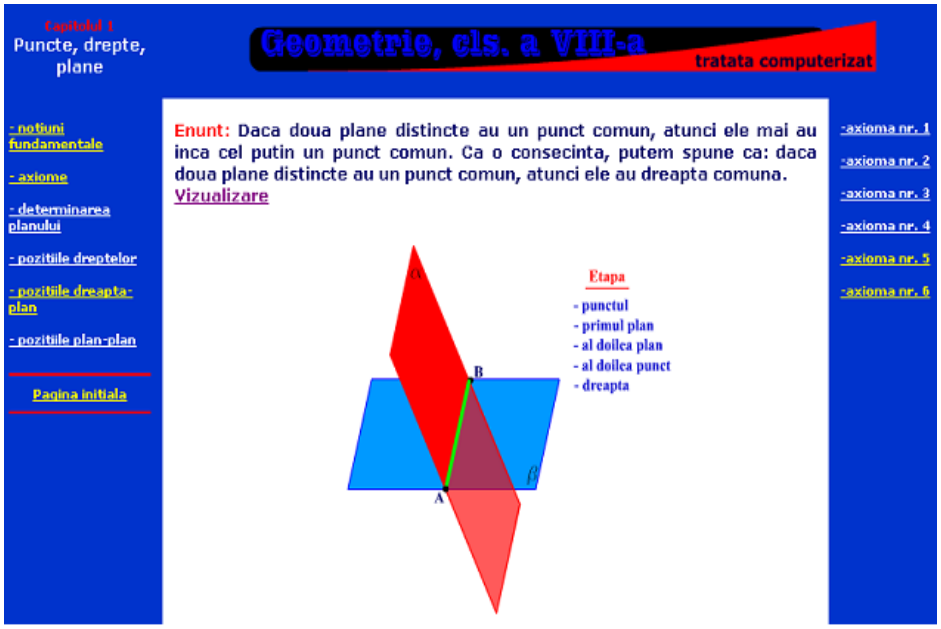


Figura 2.10.5. Poziţiile relative a două plane în Geometria 8

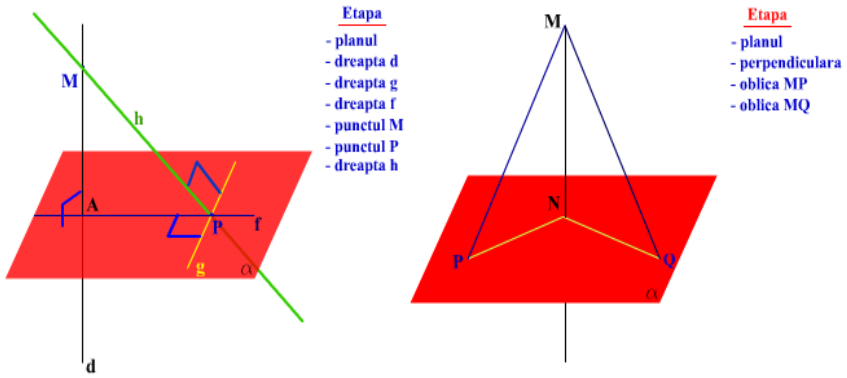


Figura 2.10.6. Teorema celor 3 perpendiculare şi distanţa de la un punct la un plan în Geometria8

- formule care fac legătura între lungimea unui segment şi proiecţia acestuia, respectiv a ariei unei figuri geometrice şi aria proiecţiei sale pe un plan.

➢ Capitolul 5. Unghiul unei drepte cu un plan, unghiul a două plane. Unghi diedru.
În acest capitol se studiază câteva tipuri noi de unghiuri.

Multiplele avantaje ale prezentărilor computerizate sunt:
- Volumul mare de informaţii pe unitatea de timp;
- Posibilitatea vizualizării figurilor geometrice utilizând diferite tehnici de animaţie;
- Modul interesant şi atractiv de prezentare a informaţiilor sau accesul la informaţii în mod gradual;
- Desenele sunt prezentate în mai multe etape;
- Materialul teoretic în program corespunde cerinţelor programei de studiu, sunt preconizate câteva niveluri de expunere în dependenţă cu nivelul de pregătire al elevilor.

Dezavantajele *software*-ului prezentat:
- Programul poate fi utilizat numai în etapa de însuşire de noi cunoştinţe; nu cuprinde opţiuni pentru etapele de consolidare a cunoştinţelor dobândite, verificare şi evaluare;
- Nu este oferită posibilitatea de rezolvare interactivă a problemelor.

2.10.3 AEL Educaţional – Lecţii de matematică

AEL [2] şi lecţiile sale au apărut ca un nou mod de a învăţa, complementar celor care s-au folosit deja. Aceste lecţii interactive ajută la formarea unei imagini mai corecte despre ceea ce se predă. Învăţarea prin descoperire a înlocuit învăţarea bazată pe memorare.

Dintre lecţiile de geometrie existente amintim cele despre cerc şi sferă. În cazul cercului se prezintă dinamic următoarele noţiuni:
- Lungimea şi aria unui cerc ;
- Raza, coarda şi diametrul unui cerc ;
- Sectorul de cerc şi segmentul de cerc.

În cazul sferei lecţia urmăreşte obiectivele (Figura 2.10.7):
- descriere, formule;
- zona sferică şi calota sferică;
- corpuri înscrise în sferă: prisma regulată dreaptă, cilindrul, piramida regulată dreaptă, conul.

Avantaje ale *software*-ului prezentat:
- Materialul teoretic în program corespunde cerinţelor programei de studiu, sunt preconizate câteva niveluri de expunere în dependenţă cu nivelul de pregătire al elevilor;
- Simplitatea lucrului cu programul.

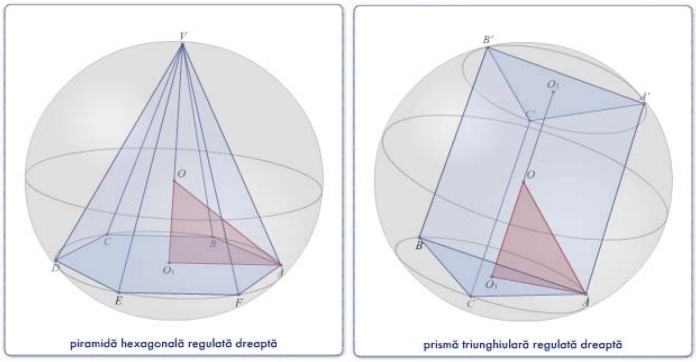


Figura 2.10.7. Corpuri înscrise în sferă în AEL Lecții de matematică

Dezavantajele *software*-ului prezentat:

- Programul poate fi utilizat numai în etapa de însuşire de noi cunoştinţe; nu cuprinde opțiuni pentru etapele de consolidare a cunoştinţelor dobândite, verificare şi evaluare;
- Lipseşte prezența componentei jocului;
- Nu sunt prezentate aplicații pentru consolidarea cunoştinţelor;
- Nu este oferită posibilitatea de rezolvare interactivă a problemelor;
- Este realizat la un nivel înalt, dar există neajunsuri metodice.

2.10.4 Geometrie, între joc şi nota 10

„Geometrie, între joc si nota 10" este un *software* educațional de geometrie care se adresează elevilor din clasele a VI-a şi a VII-a, fiind realizat de către Softwin [105]. Programul este în conformitate cu programa şcolară şi prezintă elementele fundamentale de geometrie. Introducerea noțiunilor se realizează prin intermediul unor jocuri distractive.

Cele patru produse din seria « Geometrie, între joc şi nota 10 » sunt:

- Provocarea Faraonului;
- Secretul lui Euclid;
- La voia zeilor;
- Comoara tâlharilor.

Primul produs „Provocarea Faraonului" cuprinde prima parte a materiei de clasa a VI-a şi prezintă elementele de bază ale geometriei:

- Punct, dreaptă, semidreaptă, segment:
  - Punct, dreaptă, plan;
  - Semidreaptă, semiplan;
  - Segment, segmente congruente.
- Unghiuri:
  - Definiție, notație, elemente;
  - Măsurarea unghiurilor;
  - Unghiuri congruente;
  - Clasificarea unghiurilor după măsura lor şi după poziția lor;
  - Unghiul a două drepte, perpendicularitate;
  - Drepte tăiate de o secantă, drepte paralele.
- Poligoane:
  - Poligoane concave şi convexe (Figura 2.10.8);
  - Clasificarea poligoanelor după numărul de laturi;
  - Aria şi perimetrul unui poligon.

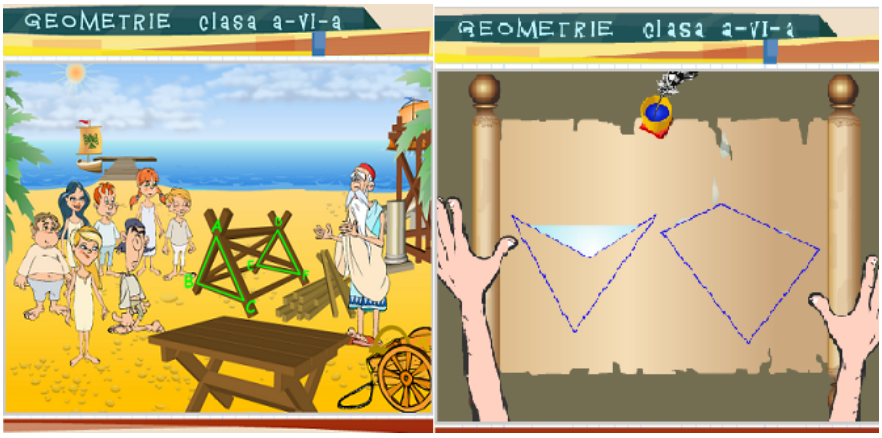


Figura 2.10.8. Triunghiuri congruente şi proprietăţi ale patrulaterelor convexe/concave în Geometrie, între joc şi nota 10

Introducerea noţiunilor se realizează prin intermediul următorului joc distractiv: în Alexandria, Egipt, anul 300 î.Hr., la palatul regal în sala tronului Euclid, părintele geometriei, propune un proiect curajos. Temutul faraon Ptolemeu I Soter îi răspunde cu o provocare. Va accepta Euclid provocarea? Va fi în stare să o ducă până la capăt? Care este secretul său?

Al doilea produs „Secretul lui Euclid” cuprinde a doua parte a materiei de clasa a VI-a şi prezintă noţiunile şi relaţiile dintr-un triunghi:

- Elementele unui triunghi;
- Clasificarea triunghiurilor;
- Construcţia triunghiurilor;
- Congruenţa triunghiurilor (Figura 2.10.8);
- Linii importante în triunghi: mediane, bisectoare, înălţimi şi mediatoare;
- Triughiul isoscel, triughiul echilateral.

Introducerea noţiunilor se realizează prin intermediul următorului joc distractiv: Euclid, părintele geometriei şi-a luat angajamentul de a construi prima şcoală din Alexandria, pentru discipolii săi. Dar oare construcţia merge conform planului? Întâmplări neprevăzute şi provocări noi din partea faraonului îi creează greutăţi marelui învăţat şi încetinesc construţia şcolii. Dar cu ajutorul cunoştinţelor sale vaste va reuşi să treacă peste piedici şi să-i înveţe pe discipoli o mulţime de lucruri noi şi interesante despre triunghi.

Al treilea produs „La voia zeilor” cuprinde prima parte a materiei de clasa a VII-a şi prezintă patrulaterele particulare:

- Paralelogram:
  - Proprietăţi ale laturilor, unghiurilor şi diagonalelor unui paralelogram;
  - Paralelograme speciale: dreptunghi, romb, pătrat;
  - Condiţii necesare şi suficiente ca un patrulater să fie paralelogram;
  - Condiţii necesare şi suficiente pentru ca un paralelogram să devină dreptunghi, romb, pătrat;
  - Centre de simetrie şi axe de simetrie.
- Trapez: definiţie, clasificare, proprietăţi;
- Arii: aria triunghiului, aria paralelogramului, aria rombului, aria trapezului.

Introducerea noţiunilor se realizează prin intermediul următorului joc distractiv: În Alexandria, acum mai bine de două milenii, Euclid şi discipolii săi au început construcţia celei dintâi şcoli de geometrie din lume. Când în sfârşit credeau că lucrurile merg bine, forţe paranormale intervin în planurile lor. Pe neaşteptate, întreaga poveste este tulburată de întâmplări stranii şi personaje misterioase.

Al patrulea produs „Comoara tâlharilor” cuprinde a doua parte a materiei de clasa a VII-a şi prezintă relaţiile metrice într-un triunghi:

- Relaţii metrice în triunghiul oarecare:
  - Segmente proporţionale;
  - Teorema lui Thales şi aplicaţiile sale.
- Relaţii metrice în triunghiul dreptunghic:
  - Proiecţii ortogonale, teorema catetei şi teorema înălţimii;
  - Teorema lui Pitagora;
  - Rapoarte constante în triunghiul dreptunghic;
  - Rezolvarea triunghiului dreptunghic.

Introducerea noţiunilor se realizează prin intermediul următorului joc distractiv: În Alexandria, capitala Egiptului, Euclid şi discipolii săi sunt pe cale de a finaliza construcţia primei şcoli de geometrie din Lumea Antică. Din păcate, acţiunile lor încurcă planurile a doi vestiţi tâlhari, iar aceştia vor face tot ce le stă în putinţă să le pună bețe în roate. Mai mult, un personaj malefic din umbră pare să-i coordoneze pe răufăcători. Inteligenţa ta şi noţiunile despre relaţiile metrice pe care le vei învăţa de la marele profesor Euclid, te vor ajuta să opreşti tâlharii! Ajută-i pe discipoli să dejoace urzelile tâlharilor şi descoperă împreună cu aceştia cine este liderul din umbră.

Avantaje ale *software*-ului prezentat:

- Materialul teoretic în program corespunde cerinţelor programei de studiu, sunt preconizate câteva niveluri de expunere în dependenţă cu nivelul de pregătire al elevilor;
- Conţinutul ştiinţific prezentat este bine structurat, existând material suplimentar;
- Programul poate fi utilizat atât în etapa de însuşire de noi cunoştinţe, cât şi în etapa de consolidare a cunoştinţelor dobândite;
- Introducerea cunoştinţelor realizându-se prin intermediul jocurilor, creşte gradul de atractivitate în utilizarea *software*-ului;
- Simplitatea lucrului cu programul.

Dezavantajele *software*-ului prezentat:

- Programul nu cuprinde opţiuni pentru etapele de verificare a cunoştinţelor şi evaluare;
- Nu este oferită posibilitatea de rezolvare interactivă a problemelor.

2.10.5 Java Applets on Mathematics

Java Applets on Mathematics realizat de către Walter Fendt [29], prezintă mai multe *applet*-uri dinamice din următoarele ramuri ale matematicii: aritmetică, algebră elementară, geometrie plană, geometrie sferică, trigonometrie, analiză matematică şi geometrie vectorială. Setul de *applet*-uri poate fi accesat în mai multe limbi de circulaţie internaţională cum ar fi: engleză, germană, franceză, italiană, spaniolă, olandeză, cehă, coreană.

La geometria plană sunt prezentate *applet*-uri pentru următoarele noţiuni:

- transformări geometrice simple;
- proprietăţi ale unghiurilor unui triunghi;
- triunghiul şi proprietăţi ale acestuia, liniile importante într-un triunghi;
- cercul înscris şi cercul circumscris unui triunghi;
- patrulatere inscriptibile şi patrulatere circumscrise (Figura 2.10.9);
- segmente proporţionale pe drepte paralele ;
- cercul şi proprietăţile acestuia (Figura 2.10.10);
- teorema lui Pitagora.

Applet-urile prezentate pot fi utilizate cu succes la orele de geometrie atât în şcoala generală, cât şi în liceu şi facultate, constituind un foarte bun material auxiliar.

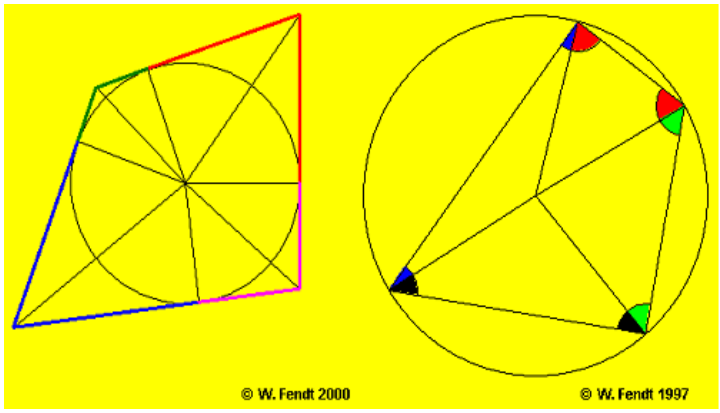


Figura 2.10.9. Patrulatere circumscrise şi patrulatere inscriptibile Java Applets on Mathematics

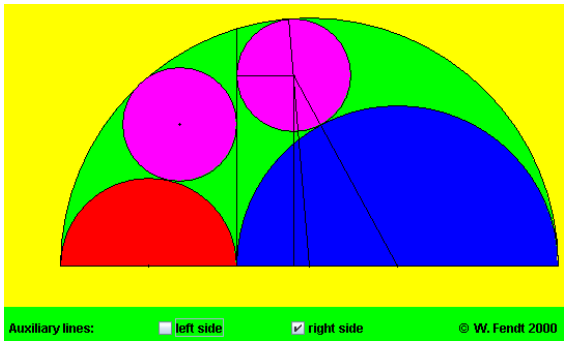


Figura 2.10.10. Cercurile lui Arhimede Java Applets on Mathematics

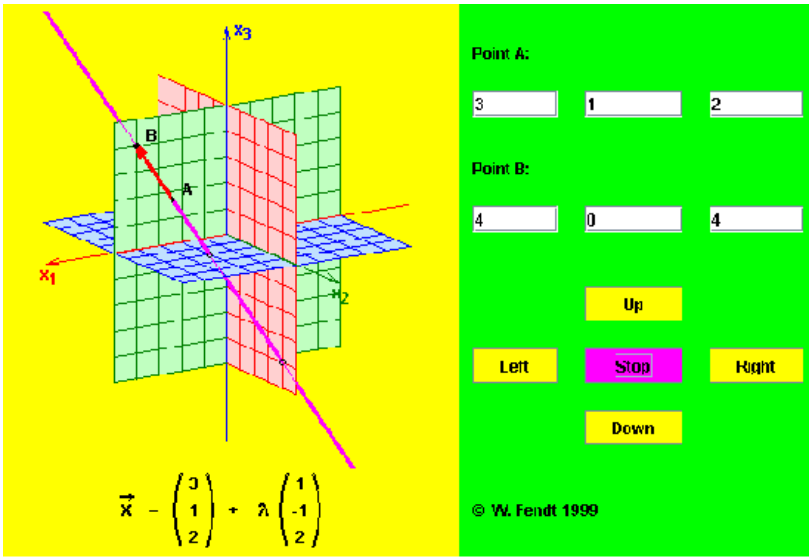


Figura 2.10.11. Ecuația vectorială a unei drepte în spațiu Java Applets on Mathematics

La geometria vectorială sunt prezentate *applet*-uri pentru următoarele subiecte:

- componentele unui vector;
- ecuația vectorială a unei drepte în spațiu (Figura 2.10.11).

Avantaje ale *software*-ului prezentat:

- Informația prezentată poate fi accesată în mai multe limbi de circulație internațională;
- Programul este preconizat pentru utilizare independentă, dar este posibilă utilizarea lui şi în procesul de instruire;
- Prezentarea noțiunilor se realizează dinamic prin intermediul *applet*-urilor.

Dezavantajele *software*-ului prezentat:

- Programul poate fi utilizat numai în etapa de însuşire de noi cunoştinţe; nu cuprinde opţiuni pentru etapele de consolidare a cunoştinţelor dobândite, verificare şi evaluare;
- Lipseşte prezenţa componentei jocului;
- Utilizarea *software*-ului este limitată doar la anumite părţi ale domeniilor abordate;
- Nu este oferită posibilitatea de rezolvare interactivă a problemelor;
- Este realizat la un nivel înalt, dar există neajunsuri metodice.

2.10.6 Cinderella

Cinderella, implementat de către Jürgen Richter-Gebert şi Ulrich Kortenkamp [62,95], este un *software* interactiv educaţional performant, uşor de utilizat, care rulează sub Windows, MacOS, Linux sau alte variante a sistemului de operare Unix. Întrucât *software*-ul este implementat în Java, se pot utiliza mai multe *browser*-e cum ar fi: Firefox, Safari, Internet Explorer. Cinderella2 reprezintă un pas major, având multe caracteristici şi dezvoltări faţă de versiunea anterioară.

Software-ul cuprinde trei mari secţiuni :

- Cinderella: interfaţă interactivă pentru realizarea construcţiilor geometrice;
- CindyScript: permite simulări ale fenomenelor fizice;
- CindyLab: limbaj de programare.

Secţiunea de geometrie « Cinderella » este proiectată pentru a acoperi un domeniu larg al disciplinelor geometrice. Se furnizează suport nativ pentru geometria euclidiană, geometria hiperbolică, geometria eliptică şi geometria proiectivă.

Interfaţa programului cuprinde o bară de meniuri, o bară cu butoane corespunzătoare celor mai importante operaţii şi suprafaţa de desenare. Dintre cele mai importante operaţii enumerăm:

- desenarea punctelor libere sau a punctelor cu anumite proprietăţi, spre exemplu mijlocul unui segment ;
- desenarea dreptelor oarecare sau a dreptelor ce îndeplinesc anumite condiţii, spre exemplu paralela la o dreaptă sau perpendiculara pe o dreaptă;
- desenarea cercurilor când se specifică centru şi raza sau când se selectează trei puncte ce vor identifica cercul;
- desenarea conicelor : parabolă, elipsă şi hiperbolă (Figura 2.10.12);
- vizualizarea unei construcţii geometrice în geometrie euclidiană, sferică sau hiperbolică, în coordonate carteziene sau polare (Figura 2.10.13);
- posibilitatea de deplasare a construcţiei geometrice.

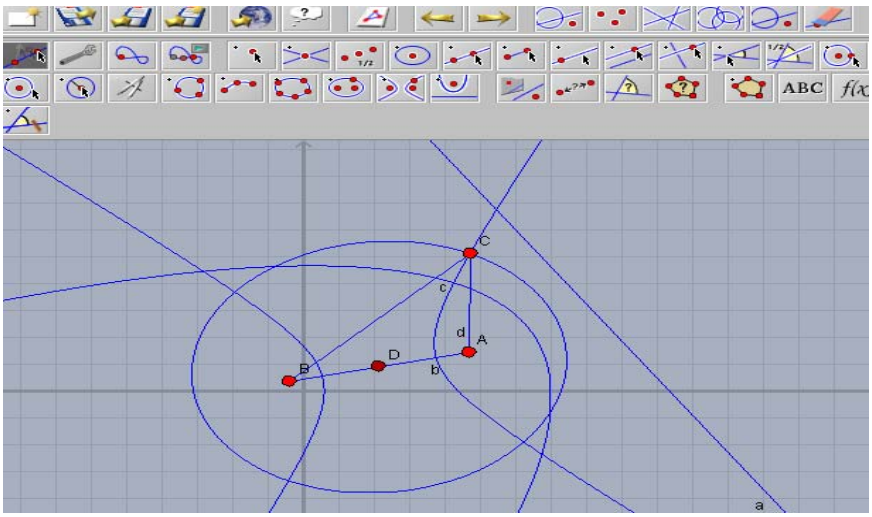


Figura 2.10.12. Conice în vizualizare euclidiană în Cinderella

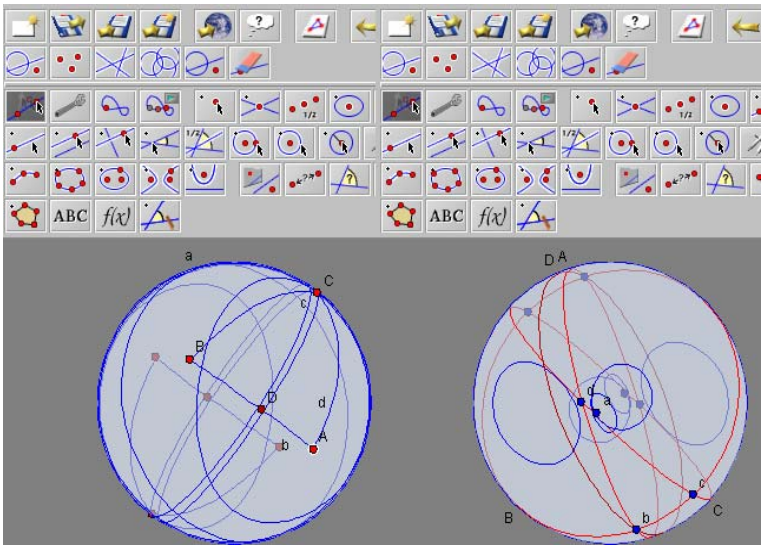


Figura 2.10.13. Conice vizualizate în geometria sferică în Cinderella

Versiunea "Cinderella 2" reprezintă un pas mare înainte față de versiunea anterioară, având multe dezvoltări ale operațiilor permise :

- desenarea poligoanelor regulate;
- posibilitatea preluărilor părților unei construcții geometrice şi reținerea în bara de butoane pentru o eventuală reutilizare;
- utilizarea conceptului de copiere/lipire/redefinire pentru a facilita reutilizarea unor construcții deja realizate ;
- recunoaşterea schițelor de mână.

« CindyScript » este un limbaj de programare funcțional care a fost proiectat pentru aplicații geometrice, dar şi pentru alte părții ale matematicii. Permite exploatarea avantajelor unui mediu geometric dinamic în conexiune cu un limbaj de programare.

Avantaje ale *software*-ului prezentat:

- Programul poate fi utilizat atât de profesor în pregătirea lecțiilor sau a cursurilor, cât şi de elevi sau studenți;
- Este oferită posibilitatea realizării de construcții geometrice precise fără utilizarea hârtiei, creionului, riglei şi a compasului;
- Programul poate fi utilizat atât în etapa de însuşire de noi cunoştințe, cât şi în etapa de consolidare a cunoştințelor dobândite;
- Desenele realizate pot fi salvate, iar schițele de mână sunt recunoscute;
- Se pot realiza aplicații pentru anumite teme prin utilizarea limbajului de programare propriu;
- Este realizat la un nivel tehnic înalt, luându-se în considerație toate cerințele metodice.

Dezavantajele *software*-ului prezentat:

- Programul nu cuprinde opțiuni pentru etapele de verificare a cunoştințelor şi evaluare;
- Utilizarea *software*-ului este o activitate complexă, de aceea se recomandă utilizarea lui în ultimii ani de liceu sau în universități.

2.10.7 Euclidraw

EucliDraw, prescurtat EUC [108], este un program care permite realizarea unor figuri din geometria euclidiană plană. Programul reprezintă un instrument modern de desenare, simulând suprafața de desenare, rigla, compasul şi oferind alte instrumente specializate, care permit desenarea figurilor complicate, instantaneu şi cu o mare precizie. Independent de utilizarea

instrumentelor, se oferă un mediu de programare în care se pot scrie şi compila instrumente proprii. Astfel se pot extinde posibilităţile programului prin crearea de colecţii de unelte specializate care permit construirea de noi figuri.

EucliDraw este rezultatul unei lungi perioade de cercetare, iniţiată în 1990 la Universitatea din Creta şi continuând până astăzi. Prima versiune statică a programului este disponibilă încă din 1998, sub numele Isoptikon.

Programul înlocuieşte hârtia cu suprafaţa ecranului calculatorului, iar creionul cu *mouse*-ul. Pentru desenarea unei figuri, prima dată trebuie ales instrumentul şi apoi utilizat *mouse*-ul. Uneori un simplu clic pe un obiect poate duce la desenarea unei figuri, spre exemplu mijlocul unui segment sau cercul circumscris unui triunghi.

Mediul dinamic permite păstrarea relaţiilor între figuri, chiar şi după modificarea unor proprietăţi ale figurilor. Astfel, de exemplu, medianele unui triunghi se redesenează corect dacă se modifică triunghiul. Pentru orice modificare a triunghiului, se observă că ele se intersectează într-un punct.

Cele mai importante opţiuni ale programului ar fi :

- posibilitatea de a utiliza figuri elementare cum ar fi: puncte, segmente, drepte, triunghiuri, patrulatere, poligoane regulate, cercuri (Figura 2.10.14) sau figuri speciale cum ar fi: triunghiuri isoscele sau dreptunghice, dreptunghiuri, romburi, pătrate, elipse, hiperbole, parabole (Figura 2.10.15);
- un catalog special, extensibil, centrat pe instrumente legate de triunghiuri;
- elemente de geometrie proiectivă;
- transformări omotetice, transformări omografice şi transformări euclidiene: translaţia, rotaţia (Figura 2.10.16);

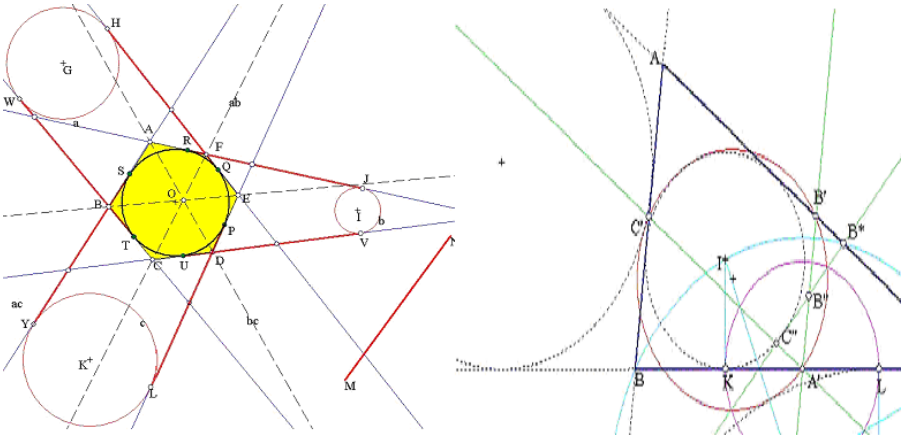


Figura 2.10.14. Teorema lui Brianchon şi teorema lui Feuerbach în Euclidraw

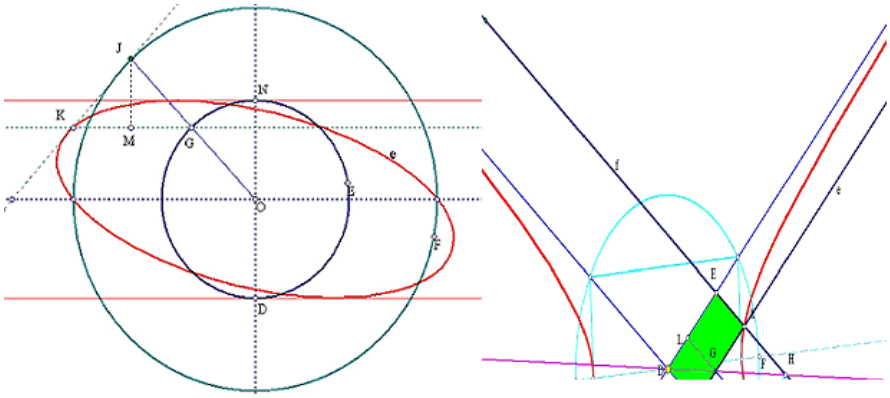


Figura 2.10.15. Construcţia unei elipse şi proprietăţi ale hiperbolei în Euclidraw

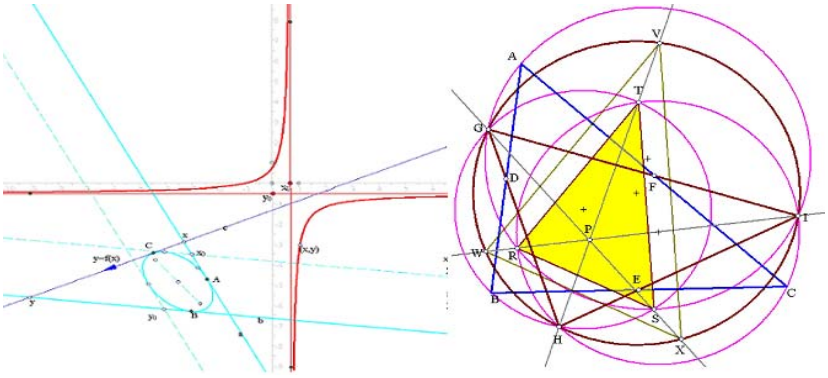


Figura 2.10.16. Exemple de transformări omografice şi euclidiene (rotaţia) în Euclidraw

- transformări arbitrare, utilizând mediul de programare;
- motoare de generare a animaţiilor şi mecanisme de mutare;
- mediul integrat de programare care permite crearea tuturor tipurilor de instrumente şi transformări;
- capacitatea de a salva construcţiile geometrice în variate formate.

Avantaje ale *software*-ului prezentat:

- Programul poate fi utilizat atât de profesor în pregătirea lecţiilor sau a cursurilor, cât şi de elevi sau studenţi;
- Este oferită posibilitatea realizării de construcţii geometrice precise fără utilizarea hârtiei, creionului, riglei şi a compasului;
- Programul poate fi utilizat atât în etapa de însuşire de noi cunoştinţe, cât şi în etapa de consolidare a cunoştinţelor dobândite;
- Se pot realiza animaţii pentru anumite desene;
- Este realizat la un nivel tehnic înalt, luându-se în consideraţie toate cerinţele metodice.

Dezavantajele *software*-ului prezentat:

- Programul nu cuprinde opţiuni pentru etapele de verificare a cunoştinţelor şi evaluare;
- Nu există posibilitatea rezolvării interactive a problemelor.

2.10.8 Geometry Applet

The Geometry Applet a fost realizat de către David E. Joyce de la Universitatea Clark, Worcester [61]. Scrierea acestor *applet*-uri a fost începută în februarie 1996. Versiunea 2.0, apărută în mai 1997, permite realizarea construcţiilor şi în spaţiul tridimensional, spre deosebire de versiunea anterioară 1.3 care permite doar realizarea construcţiilor geometrice plane.

Aceste *applet*-uri pot fi utilizate pentru a ilustra:

- elemente ale geometriei euclidiene, spre exemplu dreapta lui Euler (Figura 2.10.17);
- desenarea icosaedrului în acord cu construcţia lui Euclid (Figura 2.10.17).

„Geometry Applet”, realizat utilizând limbajul de programare Java, utilizează clasa „State Canvas” pentru a adăuga elemente pe *applet*. O instanţă a clasei „State Canvas” poate fi adăugată într-o fereastră separată, care utilizează clasa „ClientFrame”, o subclasă a clasei „Frame”.

„Elemetns” este o clasă abstractă care este clasă de bază pentru opt clase diferite corespunzătoare obiectelor ce pot fi afişate. Aceste opt clase au şi ele subclase:

- PointElement cu subclasele: AngleDivider, CircleSlider, FixedPoint, Foot, Harmonic, Intersection, IntersectionPL, InvertPoint, Layoff, LineSlider, MeanProportional, Midpoint, PlaneFoot, PlaneSlider, Proportion, Similar, SphereSlider;

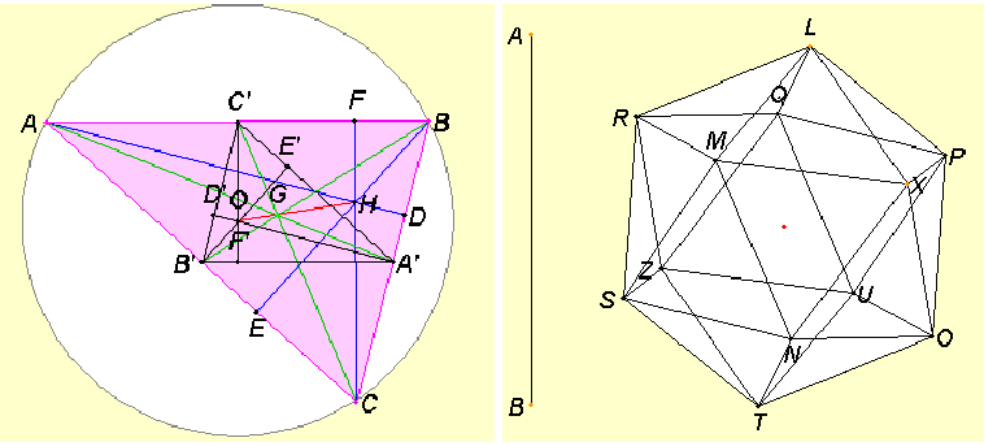


Figura 2.10.17. Dreapta lui Euler şi icosaedrul în Geometry Applet

- LineElement cu subclasele: Bichord, Chord, Perpendicular, PlanePerpendicular;
- CircleElement cu subclasele: Circumcircle, IntersectionSS, InvertCircle;
- PolygonElement cu subclasele: Application and RegularPolygon;
- SectorElement cu subclasa Arc;
- PlaneElement cu subclasele: ParallelP, PerpendicularPL, SphereElement;
- PolyhedronElement cu subclasele: Prism and Pyramid.

Avantaje ale *software*-ului prezentat:

- Programul este preconizat pentru utilizare independentă, dar este posibilă utilizarea lui şi în procesul de instruire;
- Prezentarea noțiunilor se realizează dinamic prin intermediul *applet*-urilor.

Dezavantajele *software*-ului prezentat:

- Programul poate fi utilizat numai în etapa de însuşire de noi cunoştințe; nu cuprinde opțiuni pentru etapele de consolidare a cunoştințelor dobândite, verificare şi evaluare;
- Lipseşte prezența componentei jocului;
- Utilizarea *software*-ului este limitată doar la anumite părți ale domeniilor abordate;
- Nu este oferită posibilitatea de rezolvare interactivă a problemelor;
- Este realizat la un nivel înalt, dar există neajunsuri metodice.

2.10.9 Geometria

Geometria este un *software* interactiv realizat utilizând limbajul Java, fiind distribuit de către "GeoCentral" [32]. Pentru utilizarea *software*-ului sunt necesare sistemul de operare Windows şi un *browser* web, spre exemplu: Mozilla, Internet Explorer sau Safari.

Interfața grafică confortabilă lasă să se execute acțiuni diverse asupra elementelor geometriei plane şi în spațiu. Prin utilizarea acestui *software* inteligent problemele pot fi compuse, rezolvate (Figura 2.10.18) şi trimise prin *email*.

Paleta de instrumente a *software*-ului permite operații cum ar fi:

- desenarea dreptelor, perpendicularelor, bisectoarelor, înălțimilor;
- determinarea unor măsuri: lungimi de segmente, măsuri de unghiuri, suprafețe şi volume;
- realizarea unui obiect prin compunerea mai multor obiecte (Figura 2.10.19);
- modalități diferite de vizualizare.

Software-ul interactiv „Geometria" mai cuprinde şi un pachet cu 60 de exemple clasificate, în funcție de complexitatea lor, în opt categorii.

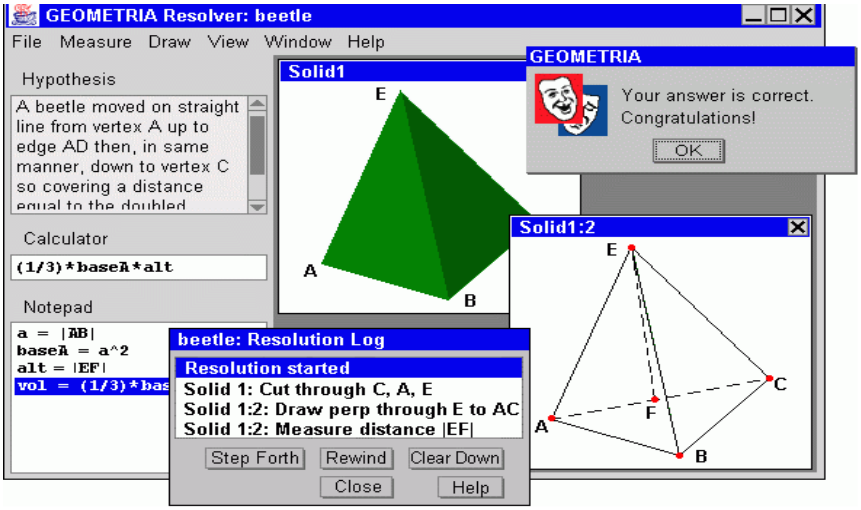


Figura 2.10.18. Interfața grafică a *software*-ului Geometria

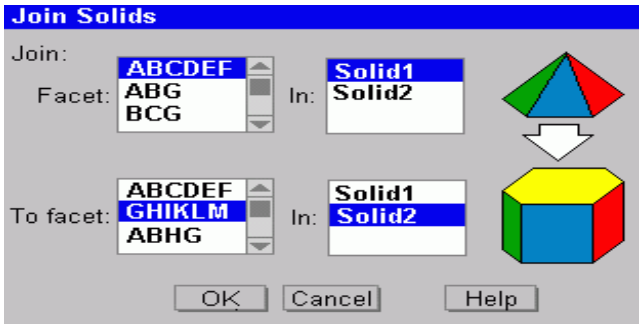


Figura 2.10.19. Realizarea unui corp solid prin compunerea unei piramide cu o prismă, bazele celor două poliedre fiind poligoane congruente în Geometria

Avantaje ale *software*-ului prezentat:

- Programul poate fi utilizat atât în etapa de însușire de noi cunoștințe, cât și în etapa de consolidare a cunoștințelor dobândite;
- Materialul teoretic în program corespunde cerințelor programei de studiu, dar există material suplimentar;
- Pentru etapa de consolidare a cunoștințelor dobândite există mai multe probleme rezolvate grupate pe opt niveluri de complexitate.

Dezavantajele *software*-ului prezentat:

- Programul nu cuprinde opțiuni pentru etapele de verificare și evaluare;
- Lipsește prezența componentei jocului;
- Este realizat la un nivel înalt, dar există neajunsuri metodice.

2.10.10 PyGeo

PyGeo este un *software* interactiv ce permite crearea de construcții geometrice dinamice, construcții ce întruchipează relații geometrice receptive în timp real la un ecran interactiv. Implementându-se un cadru abstract, este expus un domeniu de obiecte geometrice pentru construcții geometrice virtuale, dinamice. Pornind de la geometria euclidiană și metrică, scopul *software*-ului este de a dezvolta legătura dintre geometria proiectivă și spațiul real, precum și cu geometria numerelor complexe.

PyGeo este creat de către Arthur Siegel și implementat utilizând limbajul de programare Python [102], limbaj ce folosește caracteristicile programării orientate obiect, iar construcțiile

geometrice pot fi prelucrate direct prin intermediul interpretorului Python. Alegerea limbajului de programare Python este o soluție pentru scopul educațional al *software*-ului. Codul limbajului de programare ales este adesea referit ca pseudocod executabil.

Cele mai importante opțiuni ale *software*-ului sunt:

- sesiuni interactive pentru crearea construcțiilor specifice geometriei euclidiene, geometriei proiective sau geometriei numerelor complexe prin prezentarea ferestrei principale (Figura 7.20), precum şi a ferestrelor pentru detalii (Figura 2.10.21);
- crearea de construcții geometrice utilizând *prompter*-ul interactiv al limbajului de programare Python.

Avantaje ale *software*-ului prezentat:

- Programul poate fi utilizat atât de profesor în pregătirea lecțiilor sau a cursurilor, cât şi de elevi sau studenți;
- Este oferită posibilitatea realizării de construcții geometrice precise fără utilizarea hârtiei, creionului, riglei şi a compasului;
- Programul poate fi utilizat atât în etapa de însuşire de noi cunoştințe, cât şi în etapa de consolidare a cunoştințelor dobândite;
- Se pot realiza desene prin utilizarea *prompter*-ul interactiv al limbajului de programare Python;
- Este realizat la un nivel tehnic înalt, luându-se în considerație toate cerințele metodice.

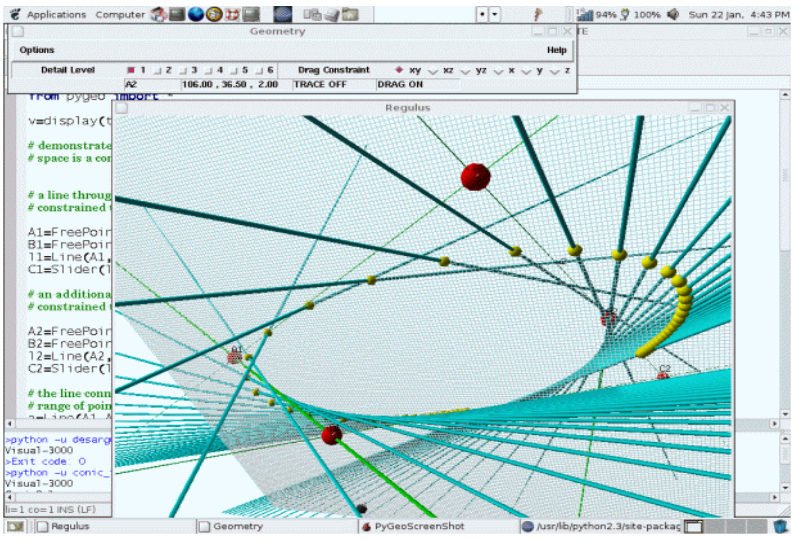


Figura 2.10.20. Sesiune interactivă în PyGeo

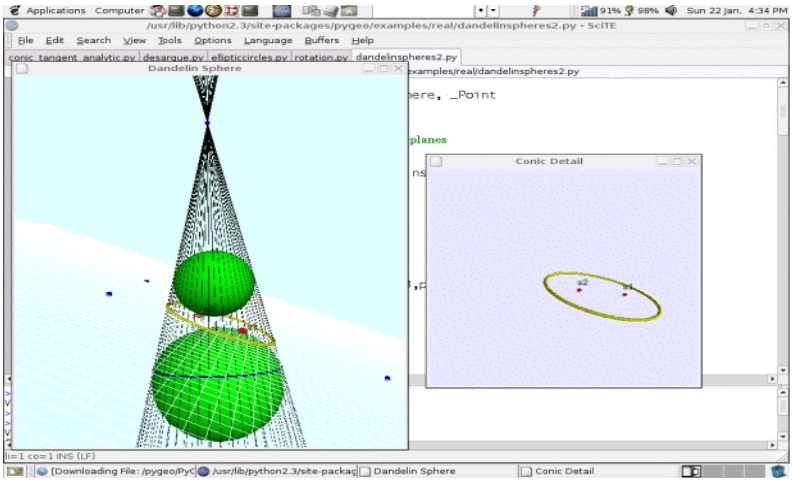


Figura 2.10.21. Sesiune interactivă cu fereastră principală şi fereastră de detaliu în PyGeo

Dezavantajele *software*-ului prezentat:

- Programul nu cuprinde opțiuni pentru etapele de verificare a cunoștințelor și evaluare;
- Nu există posibilitatea rezolvării interactive a problemelor;
- Utilizarea *software*-ului este o activitate complexă, de aceea se recomandă utilizarea lui în ultimii ani de liceu sau în universități.

2.10.11 Cabri Geometre II Plus

Cabri Geometre [21, 41], distribuit de societatea Cabrilog, a fost realizat la sfârșitul anilor '80 la Universitatea „Joseph Fourier" din Grenoble. Construcția pe calculator a figurilor geometrice aduce noi oportunități în raport cu construcțiile clasice realizate utilizând hârtia, creionul, rigla și compasul. *Software*-ul posedă un mare număr de funcționalități și facilități pentru utilizator. Figurile, de la cele mai simple până la cele mai complexe, pot fi manipulate liber.

Un document Cabri Geometre este compus dintr-o figură construită liber pe o foaie virtuală de un metru pătrat. O figură este compusă din obiecte geometrice: puncte, drepte, cercuri etc., dar, în același timp, și din alte obiecte: numere, texte, formule. De asemenea se pot memora construcții intermediare. Aplicația permite deschiderea simultană a mai multor documente și suportă mutarea sau copierea între documentele deschise.

Interfața aplicației cuprinde: bara de titlu, bara cu instrumente, bara de meniuri, bara de atribute, fereastra de descriere, bara de atribute, fereastra de ajutor, bara de stare și zona de lucru. La lansarea în execuție a aplicației bara de ajutor, fereastra de descriere și bara de atribute nu sunt vizibile. Figura 2.10.22 prezintă fereastra principală a aplicației și diferitele zone ale sale.

Cele mai importante facilități oferite de aplicație sunt:

- realizarea de construcții pornind de la obiecte geometrice elementare utilizând opțiunile oferite de instrumentele din bara de instrumente și bara de atribute [8, 65];
- efectuarea de calcule pentru determinarea unor valori precum lungimea unui segment, măsura unui unghi, aria unui poligon; de exemplu pentru determinarea ariei triunghiului prezentat se utilizează instrumentul: Measurements/Area; aria este calculată în cm^2 și poate fi afișată oriunde pe suprafața de desenare;
- vizualizarea axelor și a unui *grid* corespunzător prin utilizarea instrumentelor: Attributes/Show axes și Attributes/Define Grid;
- construcția unor locuri geometrice utilizând instrumentul: Constructions/Locus;
- reprezentarea grafică a unei funcții prin utilizarea instrumentului legat de expresii și utilizarea sistemului de axe;
- aplicarea unor transformări geometrice asupra unor obiecte deja create prin utilizarea instrumentului Transformations.

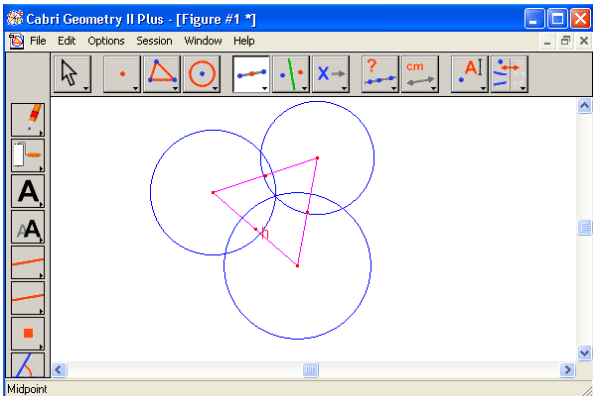


Figura 2.10.22 Fereastra principală a aplicației Cabri

Avantaje ale *software*-ului prezentat:

- Prin multitudinea de opțiuni oferite, aplicația poate înlocui cu succes hârtia şi creionul pentru realizarea construcțiilor geometrice;
- Programul poate fi utilizat la orelele de geometrie atât pentru nivelul liceal, cât şi pentru cel universitar.
- *Software*-ul este util atât în etapa de însuşire de noi cunoştințe, cât şi în etapa de consolidare a cunoştințelor dobândite;
- Este realizat la un nivel tehnic înalt, luându-se în considerație toate cerințele metodice.

Dezavantajele *software*-ului prezentat:

- Programul nu cuprinde opțiuni pentru etapele de verificare a cunoştințelor şi evaluare;
- Nu există posibilitatea rezolvării interactive a problemelor.

2.10.12 GeoGebra

GeoGebra [39] este un *software* matematic interactiv realizat de către Markus Hohenwarter de la Universitatea din Salzburg pentru utilizare în educație. *Software*-ul dinamic permite realizarea construcțiilor geometrice cu obiecte elementare precum puncte, vectori, segmente, drepte sau conice ce pot fi salvate şi modificate ulterior. Pe de altă parte, ecuațiile şi coordonatele pot fi introduse direct. GeoGebra dispune de posibilitatea utilizării de variabile pentru numere, vectori sau puncte şi de obținerea derivatei sau integralei unei funcții. Cea mai importantă caracteristică a *software*-ului este următoarea: unei expresii din fereastra corespunzătoare algebrei îi corespunde un obiect în fereastra corespunzătoare geometriei şi invers.

Dintre cele mai importante opțiuni ale instrumentelor oferite de *software* enumerăm:

- reprezentarea unui punct oarecare sau obținut prin intersecția a două drepte sau reprezentând mijlocul unui segment;
- reprezentarea unui vector determinat de două puncte sau de un punct şi un vector;
- reprezentarea unui segment determinat de două puncte sau de un punct şi o lungime;
- reprezentarea unui poligon prin precizarea punctelor; în fereastra corespunzătoare algebrei se poate vizualiza suprafața poligonului;
- reprezentarea unui cerc când se specifică trei puncte aparținând cercului sau centrul şi raza sa;
- reprezentarea unei drepte dată de două puncte, a paralelei la o dreaptă printr-un punct dat, a perpendicularei pe o dreaptă, a bisectoarei unui unghi sau a tangentei la o conică;
- reprezentarea unei conice determinată de 5 puncte, în fereastra corespunzătoare algebrei vizualizându-se aria conicei;
- utilizarea transformărilor geometrice: rotația, translația, simetria;
- inserarea de imagini şi specificarea de proprietăți;
- realizarea de animații;
- comenzii pentru a calcula arii, distanțe, integrale, măsuri ale unghiurilor etc;
- salvarea construcțiilor geometrice în diferite formate.

Avantaje ale *software*-ului prezentat:

- Prin multitudinea de facilități oferite, acest *software* matematic poate fi utilizat cu succes pentru instruirea asistată de calculator la geometrie atât în mediu preuniversitar, cât şi universitar;
- *Software*-ului este util atât în etapa de însuşire de noi cunoştințe, cât şi în etapa de consolidare a cunoştințelor dobândite;
- Existența unui editor pentru introducerea ecuațiilor sau coordonatelor;
- Permite calculul derivatelor şi integralelor;
- Este posibilă salvarea unei construcții geometrice şi modificarea ei ulterioară;
- Este realizat la un nivel tehnic înalt, luându-se în considerație toate cerințele metodice.

Dezavantajele *software*-ului prezentat:

- Programul nu cuprinde opțiuni pentru etapele de verificare a cunoştinţelor şi evaluare;
- Nu există posibilitatea rezolvării interactive a problemelor.

2.11 Concluzii

În acest capitol s-a realizat un studiu cu privire la stadiul actual în cercetările privind sistemele de instruire asistată de calculator la geometrie. S-a prezentat un scurt istoric al instruirii asistate de calculator, apoi s-a analizat rolul profesorului în învățământul actual. S-a prezentat rolul programelor de instruire asistată de calculator şi s-a propus o clasificare a acestora din punct de vedere pedagogic. În continuare au fost analizate considerentele psiho-pedagogice ale utilizării calculatorului în procesul de studiere al matematicii.

Au fost prezentate câteva contribuții personale la realizarea unor programe educaționale destinate studierii unor capitole din matematică: numere complexe, distribuția binomială, locuri geometrice elementare, fizică: legile gazului ideal, ciocniri perfect elastice şi plastice, şi informatică: metoda „Divide-et-impera", gramatici stohastice.

În încheiere sunt prezentate sistemele de instruire existente pentru geometrie, sintetizând avantajele şi dezavantajele fiecărui sistem prezentat. La etapa actuală în țară există o programatură educațională națională realizată în cadrul diferitor instituții de învățământ, precum şi o serie destul de variată de produse educaționale străine. În fiecare țară procesul de instruire are caracter național, fie prin conținutul programelor de studiu, fie prin procedeele didactice utilizate, fie chiar şi prin limba de studiu.

În orice caz, indiferent de specificul procesului de predare – învățare – evaluare național, programele educaționale moderne trebuie să posede un şir de parametri obligatorii [64,67,87]:

- Aria de aplicare a programului dat (prezentări, evaluare, autopregătire...);
- Completitudinea materialului teoretic (veridicitatea ştiințifică, corectitudinea formulării, corespunderea programului şcolar, cunoştințe istorice, prezența vocabularului);
- Existența blocului de control, metoda efectuării controlului (test, rezolvarea interactivă, rezolvarea cu introducerea răspunsului) şi alegerea problemelor;
- Posibilitatea formării de către profesor a cursului personal din materialul teoretic prezent şi administrarea lucrărilor de control cu folosirea bazei de date existente;
- Rezolvarea interactivă a problemelor, posibilitatea folosirii ajutorului la rezolvare, diagnosticarea erorilor [23,35];
- Existența în program a microcalculatorului, editorului pentru introducerea formulelor;
- Simplitatea lucrului cu programul, cu componentele separate, esteticul programului;
- Calitățile ergonomice: reprezentarea informației pe ecran (cantitatea şi calitatea informației prezentate, numărul de ferestre deschise concomitent, calitatea tiparului şi reprezentarea formulelor, evidențierea prin culori a elementelor principale);
- Prezența componentei jocului.

Fiecare dintre programele enumerate satisfac într-o măsură mai mare sau mai mică cerințele enumerate anterior [38,99], ele fiind implementate în limbaje de programare diferite şi sunt destinate pentru ramuri diferite ale geometriei (Tabelul 2.10.1).

Analizând produsele *software* prezentate se constată următoarele:

- Programele educaționale existente nu acoperă întreaga arie curriculară de studiere a matematicii, în special, în domeniul studierii geometriei prin intermediul instruirii asistate de calculator;
- Secvențele de instruire asistate de calculator trebuie intercalate de secvențe de instruire tradiționale pentru a diminua efectele negative ale influenței calculatorului asupra sănătății;

Nr. crt.	*Software*-uri educaționale	Realizatori	Limbaje de implementare	Ramuri ale geometriei
1.	Manipula Math with Java	IES Inc..	Java	-Geometrie euclidiană în plan -Geometrie analitică în plan
2.	Geometria 8	Vasile Roman	XBASE, Macromedia Flash	-Geometrie euclidiană în spațiu
3.	AEL Educațional	SIVECO România SA	JavaScript, Macromedia Flash	-Geometrie euclidiană în plan şi în spațiu
4.	Java Applets on Mathematics	Walter Fendt	Java	-Geometrie euclidiană în plan -Geometrie vectorială
5.	Cinderella	Jürgen Richter-Gebert, Ulrich Kortenkamp	Java	-Geometrie euclidiană -Geometrie proiectivă
6.	EucliDraw	Universitatea din Creta	Java	-Geometrie euclidiană
7.	The Geometry Applet	David E. Joyce	Java	-Geometrie euclidiană
8.	Geometria	GeoCentral	Java	-Geometrie euclidiană
9.	Pygeo	Arthur Siegel	Python	-Geometrie euclidiană -Geometrie proiectivă
10.	Cabri Geometre II Plus	Universitatea „Joseph Fourier” din Grenoble	Java	-Geometrie euclidiană -Geometrie analitică
11.	Geogebra	Markus Hohenwarter	Java	-Geometrie euclidiană -Geometrie analitică

Tabelul 2.10.1. Concluzii ale *software*-urilor enumerate

- În cadrul elaborării programaturii pentru studierea elementelor de geometrie este necesar de a ține cont de momentele optimale de aplicare a instruirii asistate de calculator în studierea matematicii: rezolvarea interactivă a problemelor şi evaluarea sumativă;
- Programul de instruire pentru studierea elementelor de geometrie trebuie să conțină:
  - elemente de interactivitate (simularea dialogului calculator-elev);
  - sistem de asistență pe niveluri de complexitate;
  - o bază impunătoare de probleme;
  - componente pentru evaluarea curentă şi sumativă;
  - componente pentru analiza rezultatelor.

PROIECTAREA SISTEMULUI INFORMATIC EDUCAȚIONAL

3.1 Proiectarea sistemelor informatice

3.1.1 Etapele dezvoltării sistemelor informatice

Dezvoltarea unui anumit program constă într-un set de paşi ce se fac pentru a-l realiza. Luând în considerare tipul paşilor ce se efectuează se creează un model de lucru, ce poate fi aplicat unei serii mai largi de proiecte. Acesta este motivul pentru care planul de acțiune este numit model: el poate fi privit ca un şablon al dezvoltării de programe. În timpul dezvoltării programelor s-a constatat că există anumite tipuri de activități care trebuie făcute la un moment dat [12]:

- Analiza cerințelor: Se stabileşte ce anume vrea clientul ca programul să facă. Scopul este înregistrarea cerințelor într-o manieră cât mai clară şi mai fidelă. Claritatea se referă la lipsa ambiguității, iar fidelitatea la înregistrarea cât mai exactă (posibil cuvânt cu cuvânt).
- Proiectarea arhitecturală: Din motive de complexitate, programele mari nu pot fi concepute şi implementate ca o singură bucată. Programul va trebui construit aşadar din module sau componente. Proiectarea arhitecturală împarte sistemul într-un număr de module mai mici şi mai simple, care pot fi abordate individual.
- Proiectarea detaliată: Se realizează proiectarea fiecărui modul al aplicației, în cele mai mici detalii.
- Implementarea: Proiectul detaliat este transpus într-un limbaj de programare. De obicei, aceasta se realizează modular, pe structura rezultată la proiectarea arhitecturală.
- Integrarea componentelor: Modulele programului sunt combinate în produsul final. Rezultatul este sistemul complet. În modelul numit *big-bang* componentele sunt dezvoltate şi testate individual, după care sunt integrate în sistemul final. Având în vedere că funcționarea corectă a componentelor individuale a fost testată, integrarea ar trebui să fie o formalitate. Din păcate, componentele nu pot fi testate exhaustiv, iar când acestea lucrează împreună pot să apară situații pe care o anumită componentă nu le-a întâlnit în procesul de testare sau conflicte între anumite componente (de exemplu, conflicte de partajare a resurselor). S-a constatat că atunci când se aplică acest model, timpul de testare explodează, proiectul devenind greu de controlat; aceasta justifică denumirea de *big-bang*. Modelul incremental propune crearea unui nucleu al aplicației şi integrarea a câte o componentă la un moment dat, urmată imediat de testarea sistemului obținut. Astfel, se poate determina mai uşor unde anume apare o problema în sistem. Acest tip de integrare oferă de obicei rezultate mai bune decât modelul *big-bang*.
- Validarea: În procesul de validare ne asigurăm că programul îndeplineşte cerințele utilizatorului. Un exemplu de validare este testul de acceptare, în care produsul este prezentat clientului. Clientul spune dacă este mulțumit cu produsul sau dacă mai trebuie efectuate modificări.
- Verificarea: În procesul de verificare ne asigurăm că programul este stabil şi că funcționează corect din punctul de vedere al dezvoltatorilor.
- Întreținerea: După ce programul este livrat clientului, mai devreme sau mai târziu sunt descoperite defecte sau erori ce trebuie reparate. De asemenea, pot apărea schimbări în specificațiile utilizatorilor, care vor diverse îmbunătățiri. Întreținerea constă în gestionarea acestor probleme.

Se poate constata uşor că aceste activităţi sunt în strânsă legătură cu cele patru faze ale ingineriei programării: analiza, proiectarea, implementarea şi testarea.

3.1.2 Metode de analiză orientată obiect

Obiectul reprezintă o încapsulare a valorilor unor atribute şi a serviciilor lor exclusive. O clasă descrie un set de obiecte cu atribute şi servicii comune. Un obiect este o instanţă a unei clase şi crearea unui obiect se numeşte instanţiere. Clasa poate fi descompusă în subclase. Subclasele au în comun atributele caracteristice unei familii de clase, moştenind operaţiile şi atributele claselor părinţi.

Iniţiatorii acestei metode argumentează că modul de a privi un sistem din perspectiva obiectelor este mult mai natural decât analiza sistemelor din punct de vedere funcţional. Specificaţiile bazate pe obiecte sunt mai adaptabile decât cele bazate pe funcţii.

Principiile utilizate de analiza orientată obiect: abstractizarea şi moştenirea se bazează în primul rând pe trăsăturile esenţiale ale programării orientate obiect.

Principiul abstractizării presupune ignorarea acelor aspecte ale unui subiect care nu sunt relevante pentru obiectivul curent [11]. Aceasta este o tehnică importantă pentru a coordona complexitatea. Aşadar, abstractizarea este un mecanism care permite reprezentarea unei situaţii complexe a lumii reale printr-un model simplificat. Abstractizarea procedurală este principiul conform căruia orice operaţie poate fi tratată ca o singură entitate în ciuda faptului că ea presupune realizarea mai multor operaţii de pe nivelul următor de detaliu. Această formă de abstractizare joacă un rol foarte important în definirea serviciilor sistemului.

Abstractizarea datelor reprezintă un mecanism mai puternic de abstractizare conform căruia tipurile de date sunt definite în termenii operaţiilor ce se aplică obiectelor de acel tip, cu restricţia că valorile acelor obiecte pot fi observate şi modificate doar prin intermediul operaţiilor menţionate.

Prin aplicarea acestui principiu, un analist defineşte atributele şi serviciile care manipulează aceste atribute. Singurul mod de a accesa un atribut este prin intermediul serviciilor. Atributele şi serviciile pot fi văzute ca formând un întreg. Abstracţiile sunt încapsulate în obiecte. Stările şi comportamentele comune ale obiectelor sunt încorporate în clase. Implementarea internă propriu-zisă este ascunsă de restul sistemului.

Moştenirea reprezintă mecanismul prin care se exprimă similarităţile dintre clase, simplificând definirea claselor prin utilizarea unor clase anterior definite [13]. Acest principiu pune în evidenţă generalizarea şi specializarea, făcând explicite atributele şi serviciile comune, printr-o ierarhie de clase. Principiul moştenirii permite analistului să specifice atributele şi serviciile comune doar o singură dată, sau să specializeze şi să extindă anumite atribute şi servicii.

Polimorfismul adaugă o nouă dimensiune la separarea dintre interfaţă şi implementare, dintre ce şi cum. El permite crearea de programe flexibile şi extensibile, în care noi trăsături pot fi adăugate cu uşurinţă nu numai în momentul creării proiectului, dar şi în etapele de dezvoltare ulterioare. Polimorfismul rămâne totuşi un detaliu de implementare, care poate fi ignorat în faza de analiză.

3.1.3 Procesul proiectării orientate obiect

Proiectarea este un proces creativ, care necesită o oarecare experienţă practică, acumulată în timp [78]. Acest proces implică o serie de paşi:

- Studiul şi înţelegerea problemei.

- Identificarea mai multor soluții posibile şi evaluarea fiecăreia din ele. Alegerea ei depinde de experiența proiectantului, simplitatea acesteia, valabilitatea componentelor reutilizabile.
- Descrierea fiecărei abstractizări a fiecărei soluții. Înainte de crearea documentației formale, ar putea fi necesar ca proiectantul să pregătească o descriere informativă a proiectului pentru a fi examinată în detaliu. În felul acesta, omisiunile şi erorile posibile ar putea fi eliminate înainte ca proiectul să fie documentat.

Activitățile esențiale în cursul proiectării sunt următoarele:

- Proiectarea arhitecturală: subsistemele întregului sistem sunt identificate şi documentate;
- Specificarea abstractă: pentru fiecare subsistem, se prezintă o specificare abstractă a serviciilor şi a constrângerilor sub care acestea operează;
- Proiectarea interfeței: pentru fiecare subsistem, interfața cu celelalte subsisteme este proiectată şi documentată;
- Proiectarea componentelor: serviciile furnizate de un subsistem sunt partiționate între componentele acelui subsistem;
- Proiectarea structurilor de date: structurile de date utilizate în implementarea sistemului sunt proiectate în detaliu şi specificate;
- Proiectarea algoritmilor: algoritmii utilizați pentru a furniza servicii sunt proiectați în detaliu şi specificați.

Acest proces se repetă pentru fiecare subsistem până când componentele identificate pot fi mapate direct în componentele limbajului de programare.

Continuând faza de analiză orientată obiect, proiectarea orientată obiect este o strategie în care sistemul se gândeşte în termeni de obiecte, în loc de operații şi funcții. Programul nu este proiectat ca o mulțime de funcții care schimbă date prin parametri şi memorie comună (variabile globale), ci ca o mulțime de obiecte care interacționează. Obiectele îşi păstrează starea internă şi îşi definesc operațiile pe baza acesteia. Prin ascunderea informațiilor despre reprezentarea stării, ei limitează accesul exterior la starea internă.

Aşadar, caracteristicile proiectării orientate obiect sunt următoarele:

- sistemul este proiectat ca o mulțime de obiecte care interacționează, îşi gestionează propria stare internă şi oferă servicii altor obiecte;
- obiectele sunt create prin instanțierea unei clase care defineşte atributele şi operațiile asociate obiectului; mai multe obiecte ale aceleiaşi clase pot coexista în acelaşi program;
- clasele sunt abstractizări ale lumii reale sau entități care încapsulează informații de stare şi care definesc un număr de servicii care creează, accesează sau modifică starea;
- obiectele sunt entități independente în care reprezentarea stării poate fi modificată fără a afecta alte clase din sistem;
- funcționalitatea sistemului este dată de serviciile asociate fiecărui obiect; obiectele interacționează prin apelarea serviciilor definite de alte obiecte;
- nu există zone de memorie comună; obiectele comunică prin apeluri de servicii şi nu prin variabile partajate; nu există posibilitatea ca o componentă să fie afectată de schimbarea unei informații partajate şi deci modificările sunt mai uşor de implementat;
- obiectele pot fi distribuite şi pot fi executate secvențial sau paralel; deciziile asupra paralelismului nu trebuie luate încă de la începutul procesului de proiectare.

3.1.4 Limbajul de modelare UML

Limbajul unificat de modelare UML (Unified Modeling Language) este un limbaj pentru specificarea, vizualizarea, construirea şi documentarea elementelor sistemelor *software*, însă poate fi folosit şi pentru alte sisteme, cum ar fi cele de modelare a afacerilor [79]. UML

reprezintă o colecție de practici inginereşti optime, care au fost încununate de succes în modelarea sistemelor mari şi complexe.

Dezvoltarea unui model pentru sisteme *software* industriale înainte de începerea construcției efective este esențială. Modelele bune sunt absolut necesare pentru comunicarea dintre echipele care lucrează la acelaşi proiect şi pentru asigurarea soliditătii arhitecturale. Odată cu creşterea complexitătii sistemului, creşte şi importanța unor tehnici potrivite de modelare. Există mulți factori suplimentari pentru succesul unui proiect, dar un factor esențial este respectarea riguroasă a standardelor cu ajutorul unui limbaj de modelare.

UML nu garantează succesul proiectului, dar perfecționează multe lucruri. De exemplu, scade în mod semnificativ costul instruirii în cazul schimbărilor legate de proiecte sau organizații.

Limbajul asigură posibilitatea integrării instrumentelor, proceselor şi domeniilor. Însă mai important este faptul că asigură dezvoltatorilor un mod general de rezolvare a problemelor de concepție şi planificare. Mai înainte de UML, nu exista un limbaj clar şi standardizat de modelare. Utilizatorii erau nevoiți să aleagă unul dintre multele limbaje similare, cu diferențe minore asupra puterii de expresie. Cele mai multe limbaje împărtăşeau o serie de concepte unanim recunoscute, care erau exprimate uşor diferit prin diverse notații. Aceste diferențe au fragmentat industria orientată obiect şi au descurajat noii utilizatori să învețe modelarea vizuală. De fapt, utilizatorii doreau un limbaj standardizat, o *lingua franca* a modelării.

UML 1.0 a fost propus spre standardizare în cadrul OMG (Object Management Group) în ianuarie 1997. Până la sfârşitul anului 1997 echipa care lucra la UML s-a extins, urmând o perioadă în care UML a primit o specificare formală mai riguroasă. Versiunea UML 1.1 a fost adoptată ca standard de catre OMG în noiembrie 1997. În martie 2003 a fost publicată versiunea 1.5.

În UML există numeroase diagrame (modele), aceasta favorizând existența mai multor puncte de vedere privind sistemul. Aceste diagrame pot fi clasificate astfel:

- Faza de analiză:
  - Diagrama cazurilor de utilizare;
  - Diagrama de activități;
- Faza de proiectare:
  - Structura:
    - Diagrama de clase;
    - Diagrama pachetelor;
  - Comportamentul:
    - Diagrama de stări;
    - Diagrama de interacțiuni;
      - Diagrama de secvențe;
      - Diagrama de colaborare;
- Faza de implementare:
  - Diagrama de componente;
  - Diagrama de lansare.

Diagrama cazurilor de utilizare este un instrument UML foarte puternic pentru reprezentarea cazurilor de utilizare, adică descrierea mulțimii de interacțiuni dintre utilizator şi sistem. Prin construirea unei colecții de diagrame de cazuri de utilizare, putem descrie întregul sistem într-o manieră clară şi concisă.

Diagrama cazurilor de utilizare poate conține [97]:
- Cazuri de utilizare: funcționalități ale sistemului;
- Actori: entități externe cu care sistemul interacționează;

- Relații.

Cazul de utilizare:

- este o descriere a unei mulțimi de scevențe de acțiuni (incluzând variante) pe care un program le execută atunci când interacționează cu entitățile din afara lui şi care conduc la obținerea unui rezultat observabil;
- poate fi un sistem, un subsistem, o clasă, o metodă;
- reprezintă o funcționaliatate a sistemului;
- precizează ce face un program sau un subprogram;
- nu precizează cum se implementează o funcționalitate;
- identificarea cazurilor de utilizare se face pornind de la cerințe ale clientului şi analizând descrierea problemei.

Diagramele de activități sunt folosite pentru modelarea proceselor sau a algoritmilor din spatele unui anumit caz de utilizare [28]. Din multe puncte de vedere, diagrama de activități din UML este echivalentul orientat obiect al diagramei fluxurilor de date din dezvoltarea structurată.

Modelarea conceptuală (numită şi modelarea domeniului) este activitatea de identificare a conceptelor importante pentru sistem [9]. În tehnica de programare orientată obiect, modelarea conceptuală se realizează prin diagrama claselor, întrucât clasele reprezintă concepte. Diagrama claselor furnizează structura codului care va fi scris. Problema principală este identificarea conceptelor.

Entitățile UML pot fi grupate în pachete, containere logice în care pot fi plasate elemente înrudite, ca şi directoarele din sistemele de operare [98]. Deşi orice entitate UML poate fi introdusă într-un pachet, de obicei rolul pachetelor este de a grupa clase şi uneori cazuri de utilizare înrudite.

Într-un pachet UML numele elementelor trebuie să fie unice. Totuşi, un avantaj important al pachetelor este că mai multe clase pot avea acelaşi nume dacă aparțin unor pachete diferite. Dacă două echipe *A* şi *B* lucrează în paralel, echipa *A* nu va trebui să se preocupe de conținutul pachetului echipei *B*, cel puțin din punctul de vedere al denumirilor. Aşadar, utilitatea pachetelor apare deoarece elementele sistemelor mari pot fi grupate în subsisteme mai mici şi este permisă dezvoltarea iterativă în paralel.

Diagramele de stări UML descriu diferitele stări în care se poate găsi un obiect şi tranzițiile dintre aceste stări [111]. O stare reprezintă o etapă în modelul comportamental al unui obiect şi, la fel ca în cazul diagramelor de activități, este posibil să avem stări inițiale şi stări finale. O stare inițială este cea în care se găseşte obiectul când este creat. O stare finală este o stare din care nu mai există tranziții. Tranziția reprezintă schimbarea stării, trecerea dintr-o stare în alta, şi poate fi determinată de un eveniment extern sau intern.

Diagrama de secvențe pune accentul pe aspectul temporal (ordonarea în timp a mesajelor), fiind potrivită specificațiilor de timp real şi scenariilor complexe [9]. Notația grafică este un tabel care are pe axa *X* obiecte, iar pe axa *Y* mesaje ordonate crescător în timp. Axa *Y* arată pentru fiecare obiect timpul ca o linie verticală punctată şi perioada în care obiectul preia controlul execuției (reprezentată printr-un dreptunghi) şi efectuează o acțiune, direct sau prin intermediul procedurilor subordonate.

Diagrama de colaborare se concentrează pe rolurile instanțelor şi relațiile dintre ele. Ea nu conține timpul ca o dimensiune separată, de aceea secvența de comunicații şi firele de execuție concurente trebuie numerotate. Poate conține obiecte, clase, actori, legături între acestea şi mesaje.

Diagrama componentelor este asemănătoare cu diagrama pachetelor, permițând vizualizarea modului în care sistemul este divizat şi a dependențelor dintre module [78].

Diagrama componentelor pune însă accentul pe elementele *software* fizice (fişiere, biblioteci, executabile) şi nu pe elementele logice, ca în cazul pachetelor.

Diagramele de lansare descriu configuraţia elementelor de prelucrare la *run-time* şi componentele *software*, procesele şi obiectele care se execută pe ele. Aceste diagrame sunt grafuri de noduri conectate de asociaţii de comunicare. Nodurile pot conţine instanţe ale componentelor, indicând faptul că acea componentă rulează sau se execută în nodul respectiv. Nodurile sunt reprezentate prin paralelipipede. Cercurile reprezintă interfeţe.

3.2 Descrierea sistemului informatic educaţional

Sistemul informatic, înglobând metode şi tehnici moderne, va conduce subiectul care îl utilizează la obţinerea unei experienţe în înţelegerea şi stăpânirea de cunoştinţe din domeniul geometriei şi va oferi accesul comod şi eficient la informaţiile şi cunoştinţele cele mai noi.

În realizarea sistemului informatic interactiv destinat studiului geometriei s-a urmărit atingerea următoarelor scopuri:

- prezentarea noţiunilor teoretice şi a principalelor rezultate;
- prezentatea interactivă de aplicaţii pentru fiecare subdomeniu solicitat;
- realizarea de desene exacte prin înlocuirea creionului şi a riglei cu *mouse*-ul.

Programul de prezentare de noi cunoştinţe va prezenta materialul pe baza unui dialog de investigare, interacţiunea fiind controlată de către utilizator. Investigaţia poate fi orientată spre atingerea unor scopuri precise sau poate fi o explorare. Pe lângă prezentarea noilor cunoştinţe, sunt concepute probleme rezolvate prin utilizarea teoremelor şi relaţiilor prezentate.

Interfaţa grafică utilizată pentru realizarea exactă a desenelor în plan şi în spaţiu va cuprinde o bară de meniuri şi o bară de butoane ce vor oferi instrumentele necesare pentru obţinerea exactă a desenului dorit, precum şi suprafaţa de desenare. Dintre cele mai importante opţiuni amintim:

- desenarea punctelor libere sau a punctelor cu anumite proprietăţi, cum ar fi: mijlocul unui segment, centrul de greutate al unui triunghi, ortocentrul unui triunghi, punctul lui Lemoine, intersecţia a două drepte neparalele, normala la elipsoid într-un punct, intersecţia unei drepte cu un plan etc.;
- desenarea dreptelor determinate de două puncte, a dreptelor determinate de un punct şi pantă sau a dreptelor ce îndeplinesc anumite condiţii, cum ar fi: paralela la o dreaptă ce trece printr-un punct, suportul unui segment, mediatoarea unui segment, tangenta şi normala la o conică printr-un punct selectat etc.;
- desenarea semidreptelor determinate de două puncte sau a semidreptelor determinate de anumite proprităţi, cum ar fi bisectoarea interioară a unui unghi etc.;
- desenarea segmentelor determinate de două puncte sau a segmentelor a căror extremităţi sunt puncte particulare, cum ar fi: mediana sau simediana într-un triunghi corespunzătoare unui vârf, diagonala unui patrulater determinată de un vârf, diametrul unui cerc, linia mijlocie într-un trapez etc.;
- desenarea vectorilor determinaţi de origine şi extremitate, a vectorilor de poziţie, vectorului director al unei drepte, vectorul normal al unui plan etc.;
- desenarea unghiului determinat de trei puncte sau a unghiului determinat de două semidrepte;
- desenarea planului determinat de trei puncte, a planului determinat de două drepte, a planului determinat de cei patru coeficienţi sau a unui plan determinat prin anumite proprietăţi, cum ar fi: planul ce trece printr-un punct dat şi este perpendicular pe o dreaptă, planul tangent la sferă într-un punct;
- desenarea poligoanelor oarecare şi a poligoanelor regulate;

- desenarea conicelor determinate de cinci puncte şi a conicelor particulare: elipsă, cerc, hiperbolă şi parabolă;
- desenarea cuadricelor: elipsoid, hiperboloid, paraboloid, sfera;
- desenarea pseudosferei şi a elicoidului;
- aplicarea de transformări geometrice.

Pentru elaborarea sistemului educaţional informatic se vor utiliza ramuri ale informaticii precum: programarea orientată pe obiect, teoria algoritmilor şi ingineria *software*, iar ca limbaj de programare se va folosi limbajul Java [10,66,103].

Utilizarea sistemului informatic în studierea geometriei va contribui la formarea şi dezvoltarea culturii informaţionale a elevilor. Instruirea asistată de calculator în procesul de studiu al elementelor de geometrie este şi o metodă eficientă de sporire a motivaţiei învăţării acestei discipline şi a calităţii însuşirii ei.

3.3 Faza de analiză

3.3.1 Diagrama cazurilor de utilizare

Utilizând limbajul de modelare UML [15], analiza unui sistem informatic constă în realizarea diagramei cazurilor de utilizare şi a diagramelor de activităţi. Pentru realizarea diagramelor se va folosi utilitarul ArgoUML.

Sistemul informatic va fi descris într-o manieră clară şi concisă prin reprezentatea cazurilor de utilizare [97]. Fiecare caz descrie interacţiuni între utilizator şi sistem. Reprezentarea diagramei cazurilor de utilizare este prezentată în figura 3.3.1.

Diagrama prezentată defineşte domeniul sistemului, permiţând vizualizarea dimensiunii şi sferei de acţiune a întregului proces de dezvoltate. Aceasta conţine:

- un actor – utilizatorul care reprezintă entitatea externă cu care sistemul interacţionează;
- patru cazuri de utilizare ce descriu funcţionalităţile sistemului;
- relaţii între utilizator şi cazurile de utilizare (relaţii de asociere), precum şi relaţii între cazurile de utilizare (relaţii de dependenţă).

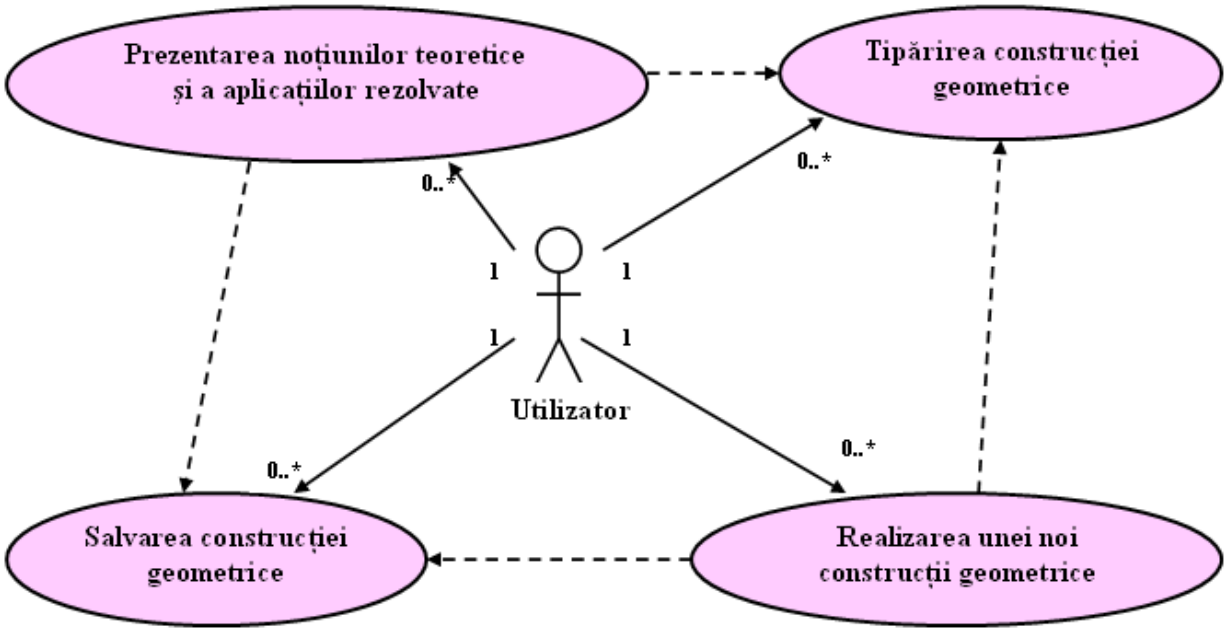


Figura 3.3.1. Diagrama cazurilor de utilizare

3.3.2 Diagramele de activităţi

Pentru fiecare caz de utilizare prezentat în diagrama anterioară s-au construit câte o diagramă de activităţi [28]. Fiecare diagramă precizează procesele sau algoritmii care sunt în spatele cazului de utilizare analizat. În următoarele patru figuri sunt prezentate aceste diagrame.

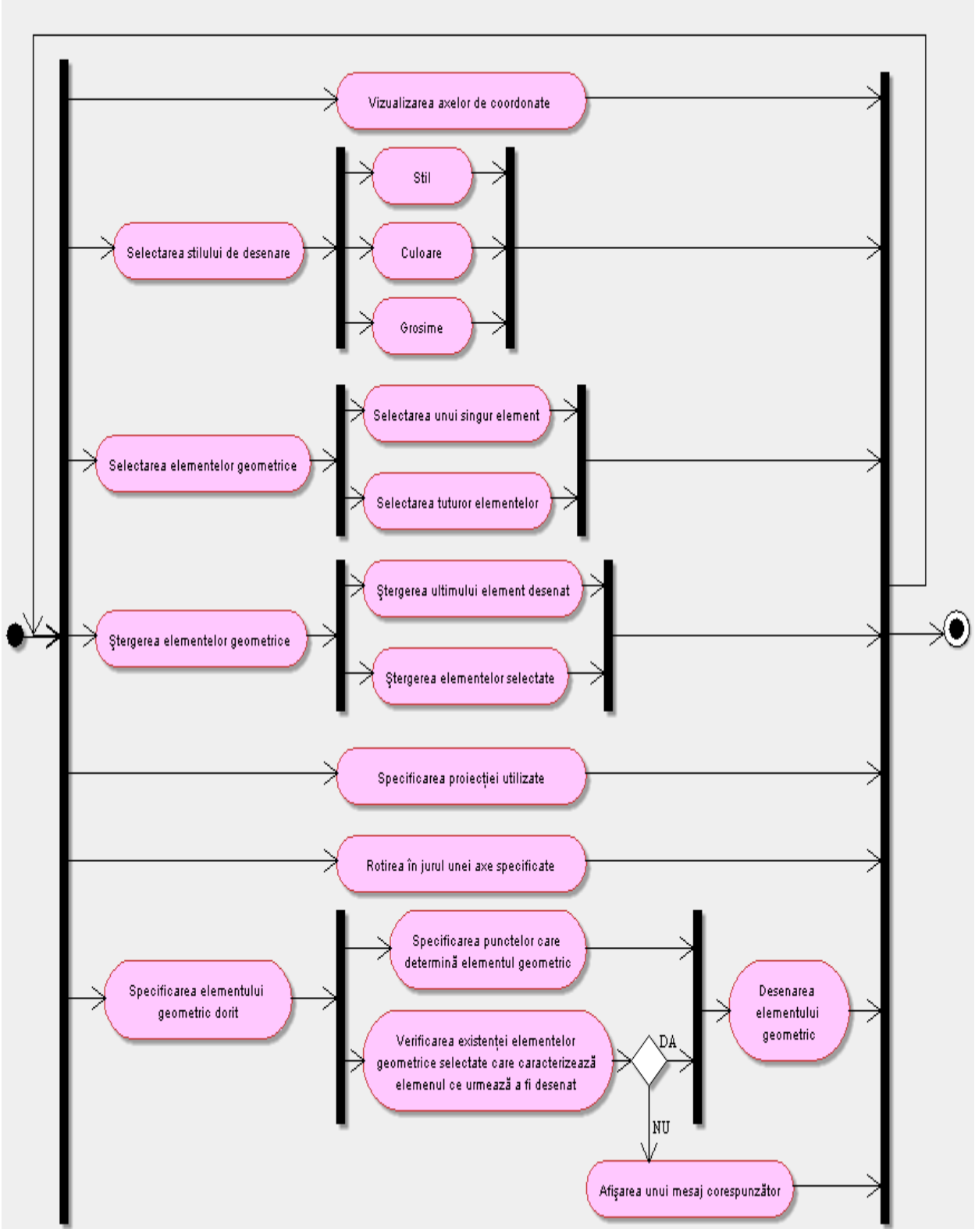


Figura 3.3.2. Diagrama de activităţi corespunzătoare cazului de utilizare:
Realizarea unei noi construcţii geometrice

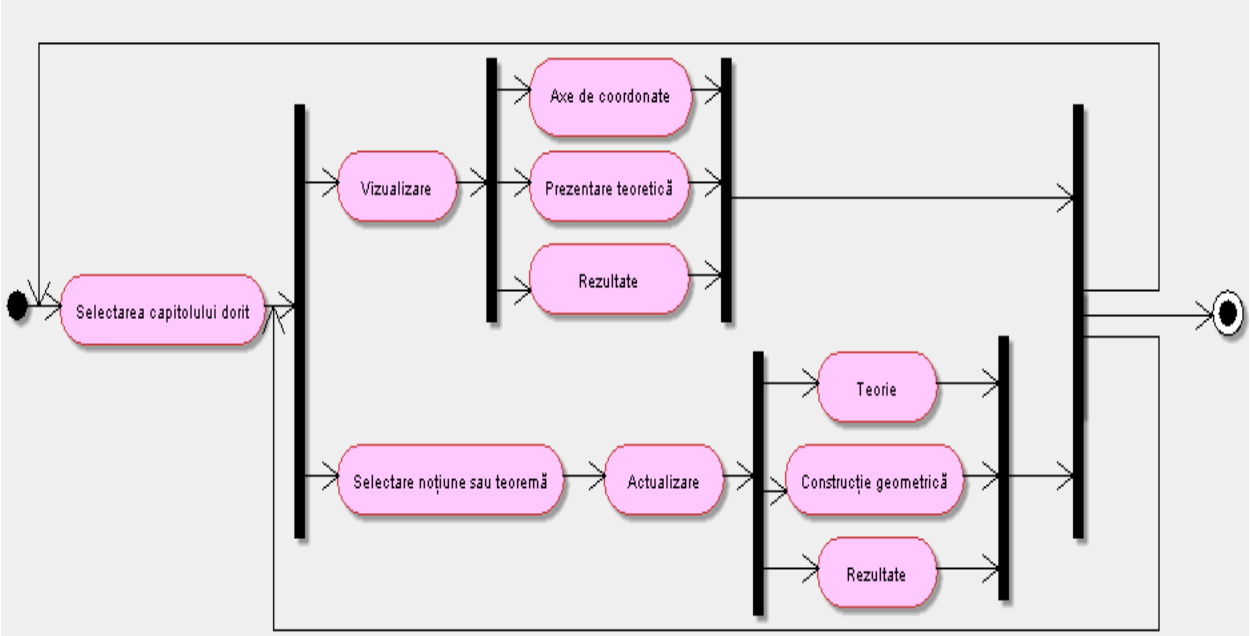


Figura 3.3.3. Diagrama de activități corespunzătoare cazului de utilizare:
Prezentarea noțiunilor teoretice şi a aplicațiilor rezolvate

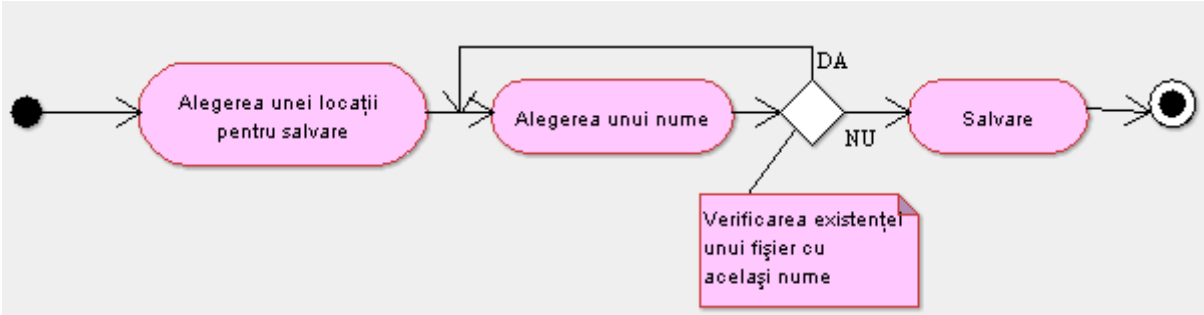


Figura 3.3.4. Diagrama de activități corespunzătoare cazului de utilizare:
Salvarea contrucției geometrice

Figura 3.3.5. Diagrama de activități corespunzătoare cazului de utilizare:
Tipărirea contrucției geometrice

Cele patru diagrame de activități reprezentate sunt folosite pentru a vizualiza, specifica, construi şi documenta aspecte dinamice legate de procesele sistemului informatic. Acestea pun accentul pe fluxul de control care urmăreşte trecerea, într-o anumită ordine, de la o activitate la alta.

3.4 Faza de proiectare

3.4.1 Diagrama claselor

Modelarea conceptuală permite identificarea celor mai importante concepte pentru sistemul informatic. Întrucât clasele reprezintă concepte [31], vom prezenta în continuare clasele ce se vor utiliza pentru identificarea elementelor geometrice.

- Grupul *geometrie în plan* va fi format din următoarele clase:
  - Element2D;
  - Punct2D;
  - Dreapta2D, Segment2D;
  - Semidreapta2D, Semiplan2D;
  - Unghi2D, Vector2D;
  - Poligon2D, Poligon_regulat2D;
  - Triunghi2D, Triunghi_echilateral2D;
  - Patrulater2D;
  - Trapez2D;
  - Paralelogram2D, Dreptunghi2D;
  - Romb2D, Patrat2D;
  - Cerc2D;
  - Conica2D;
  - Elipsa2D, Hiperbola2D, Parabola2D;
  - Izometrie2D;
  - Simetrie2D, Translatie2D, Rotatie2D;
  - Omotetie2D, Inversiune2D.
- Grupul *geometrie în spațiu* va fi format din următoarele clase:
  - Element3D;
  - Punct3D;
  - Dreapta3D, Segment3D;
  - Vector3D;
  - Plan3D;
  - Elipsoid, Hiperboloid, Paraboloid;
  - Sfera, Pseudosfera, Elicoid;
  - Izometrie3D;
  - Simetrie3D, Translatie3D, Rotatie3D.

Pe lângă clasele corespunzătoare elementelor geometrice, se vor mai folosi clase pentru realizarea construcțiilor geometrice, a interfeței pentru prezentarea noțiunilor şi a teoremelor şi a interfeței pentru suprafața de desenare.

Utilizând notația UML a unei clase, în anexa 1 se va descrie fiecare clasă prezentată anterior.

3.4.2 Diagrame de relații între clase

Clasele din arhitectura prezentată în anexa 1 au atribute şi metode comune. Moştenirea nu s-a folosit decât ca mecanism de generalizare, adică atunci când clasele derivate sunt specializări ale clasei de bază [78]. În figura 3.4.2 se poate observa că toate clasele derivate din clasa *Poligon* sunt cazuri particulare ale obiectelor acestei clase de bază.

Toate definiţiile clasei de bază se aplică tuturor claselor derivate. De exemplu, în figura 3.4.5 se poate observa că toate atributele şi metodele clasei *Elipsoid* se vor aplica atât clasei derivate *Sfera*, cât şi clasei *Pseudosfera* derivată indirect din aceasta. În următoarele şase figuri vom prezentata relaţiile de moştenire existente între aceste clase prin intermediul diagramelor de relaţii între clase.

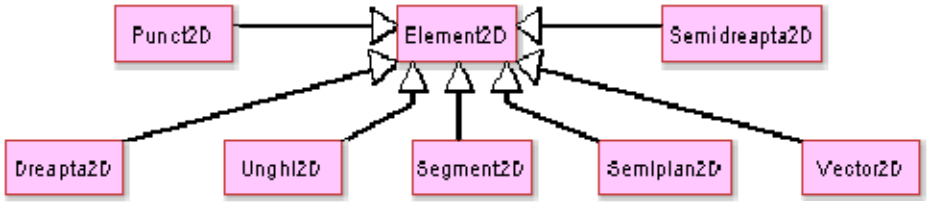


Figura 3.4.1. Diagrama ce prezintă relaţia de moştenire între elemente geometrice în plan

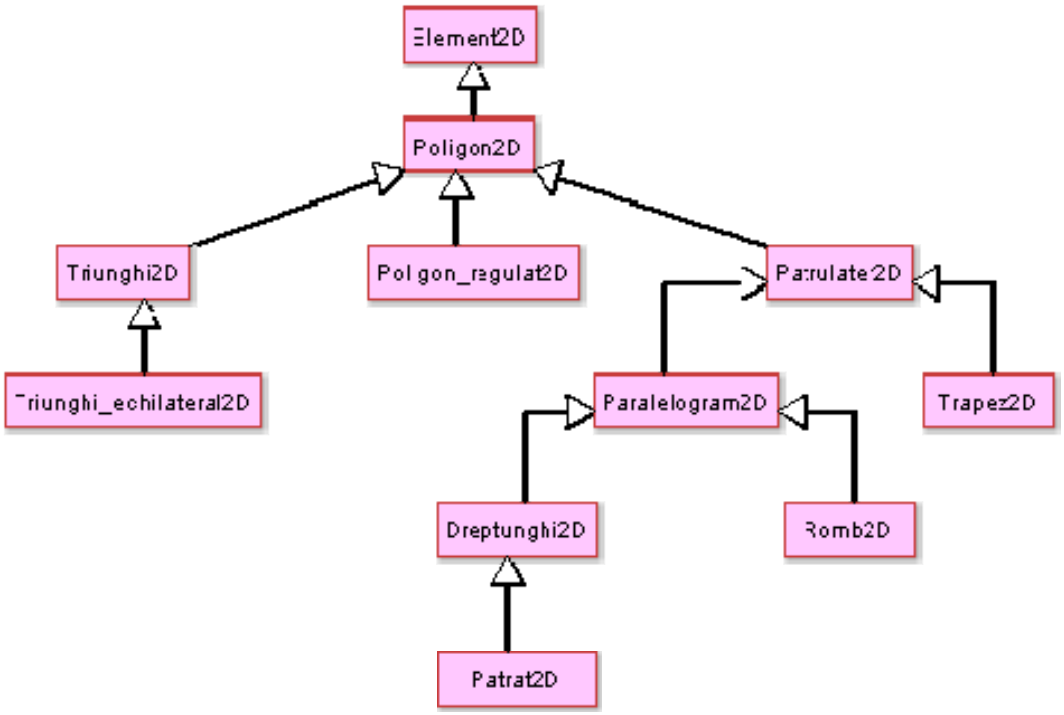


Figura 3.4.2. Diagrama ce prezintă relaţia de moştenire între poligoane definite în plan

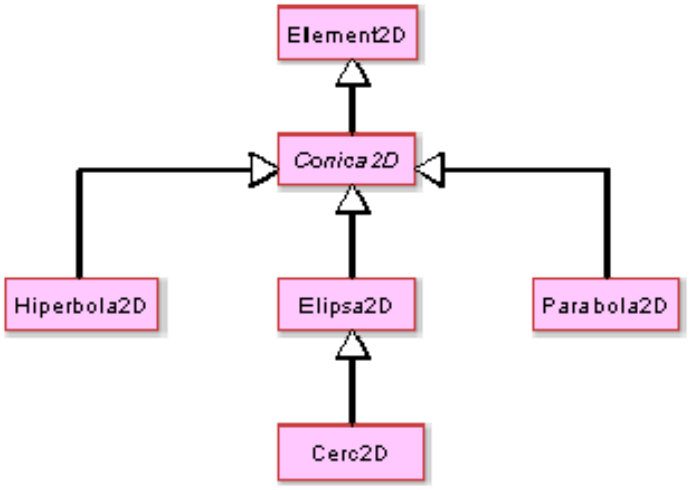


Figura 3.4.3. Diagrama ce prezintă relaţia de moştenire între conice definite în plan

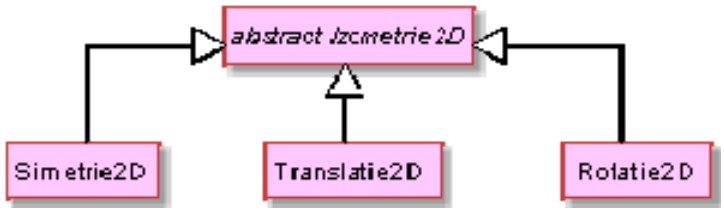


Figura 3.4.4. Diagrama ce prezintă relația de moştenire între izometriile definite în plan

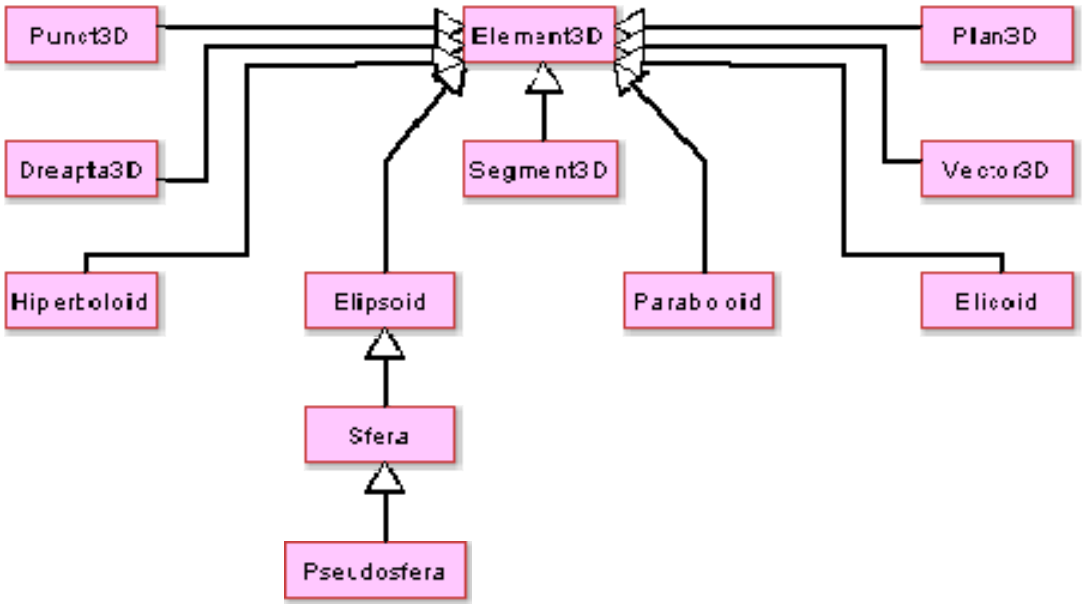


Figura 3.4.5. Diagrama ce prezintă relația de moştenire între elemente geometrice în spațiu

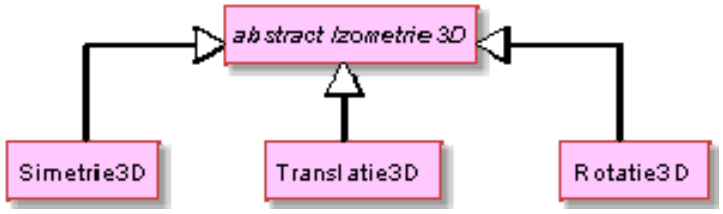


Figura 3.4.6. Diagrama ce prezintă relația de moştenire între izometriile definite în spațiu

3.4.3 Diagrame de relații între instanțe ale claselor

Între instanțele claselor din arhitectura prezentată în anexa 1 există în special relații de compunere şi de agregare [98]. În următoarele şapte figuri vom prezenta aceste relații prin intermediul diagramelor de relații între instanțe ale claselor.

Relația de agregare este o asociere în care se specifică cine este întregul şi cine este parte. Analizând figura 3.4.7 se poate observa că un obiect de tip *SuprafațaDesen2D* sau de tip *SuprafataDesen3D* reprezintă o parte a unui obiect de tip *Geometrie*. Într-o astfel de relație este posibil ca un obiect să aparțină la mai multe instanțe întreg. Spre exemplu obiectul de tip *Element2D* poate să aparțină, conform figurii 3.4.7, instanțelor de tip *Desen2D*, *DesenTriunghi*, *DesenPatrulater* şi *DesenVector*.

În cazul relației de compunere, față de agregare, instanța întreg nu poate exista fără obiectele parte. Analizând figura 3.4.8 se poate observa că o instanță de tip *Segment2D* este

constituită din două obiecte de tip *Punct2D* (extremitățile segmentului) şi un obiect de tip *Dreapta2D* (suportul segmentului).

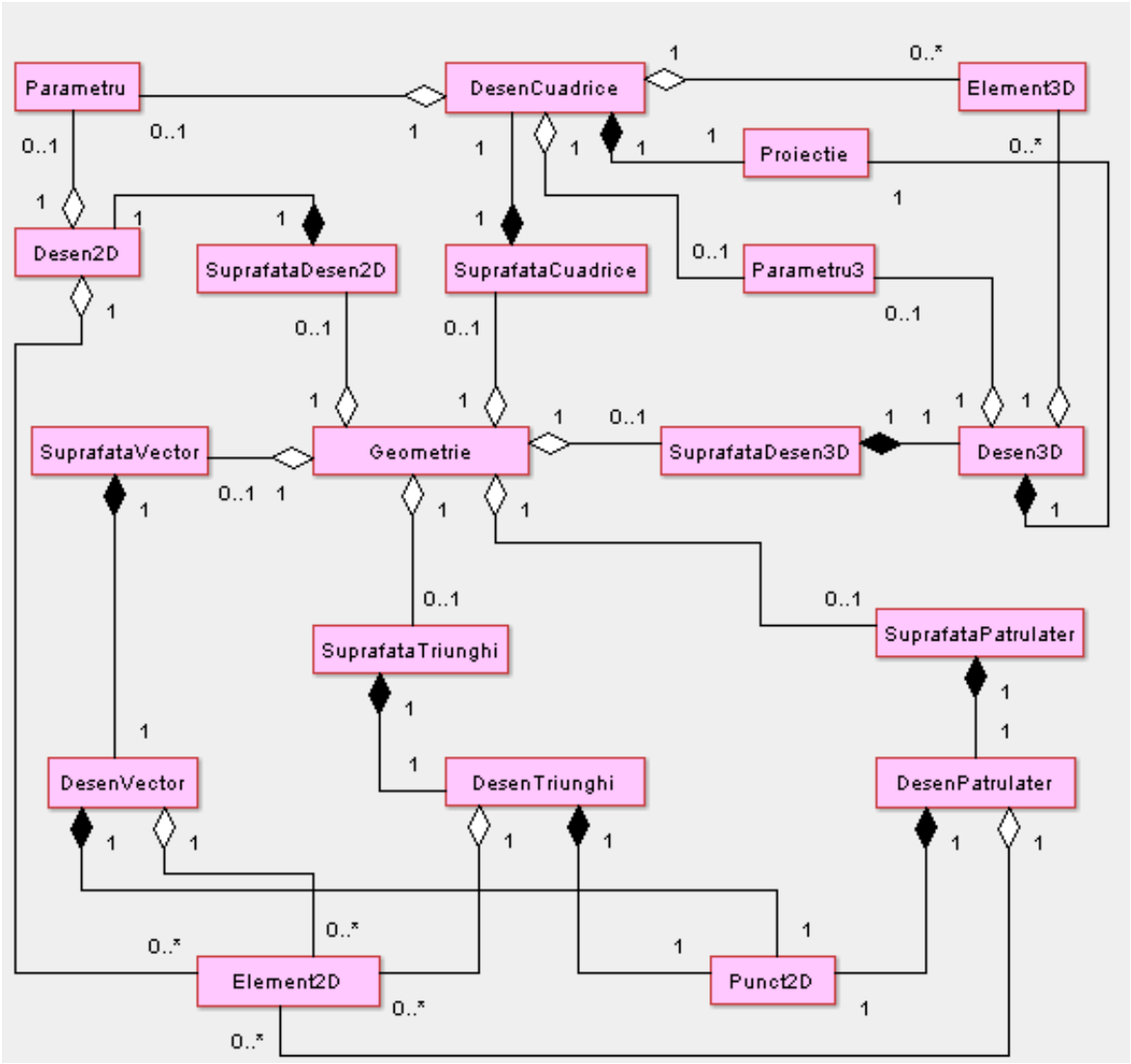


Figura 3.4.7. Diagrama ce prezintă relații de compunere şi de agregare între obiectele care permit realizarea construcțiilor geometrice

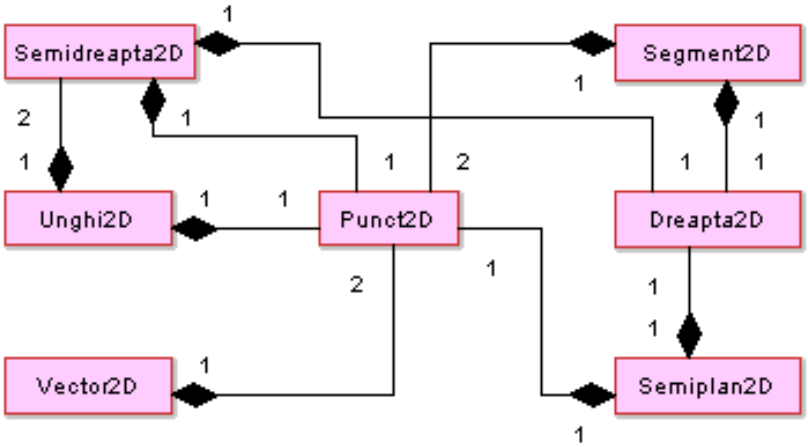


Figura 3.4.8. Diagrama ce prezintă relații de compunere între obiecte geometrice elementare în plan

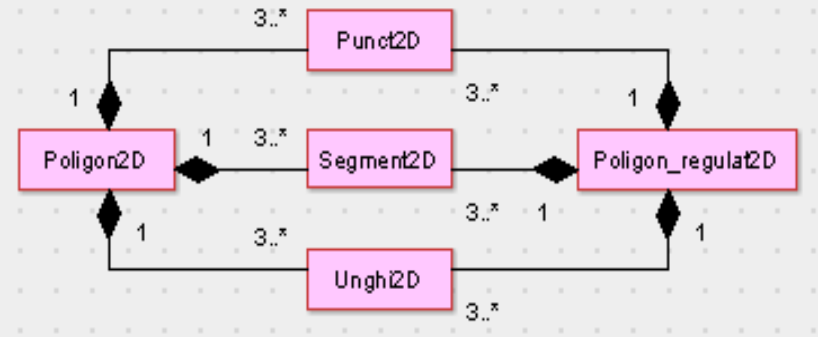


Figura 3.4.9. Diagrama ce prezintă relaţii de compunere
între elementele geometrice ce caracterizează poligoanele în plan

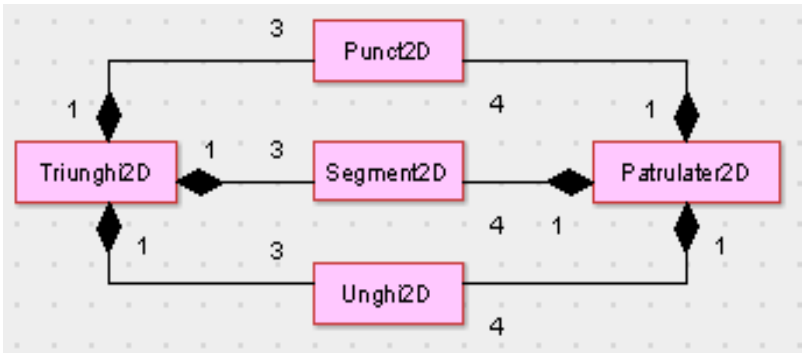


Figura 3.4.10. Diagrama ce prezintă relaţii de compunere
între elementele geometrice ce caracterizează triunghiul şi patrulaterul în plan

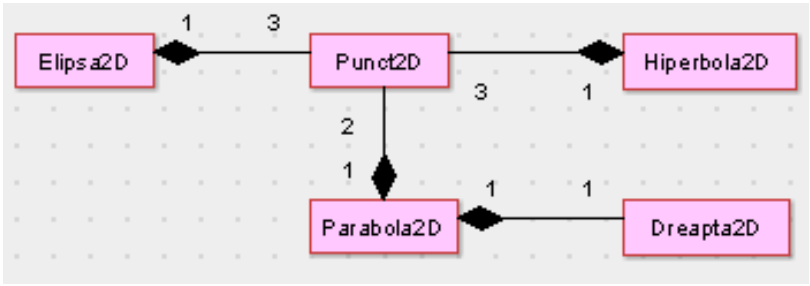


Figura 3.4.11. Diagrama ce prezintă relaţii de compunere între elementele
geometrice ce caracterizează conicele în plan

Analizând figura 3.4.9 se poate observa că redă doar relaţii de compunere. O instanţă a clasei *Poligon2D* este determinată de „n” obiecte de tip *Punct2D* (vârfurile poligonului), „n” obiecte de tip *Segment2D* (laturile poligonului) şi „n” obiecte de tip *Unghi2D* (unghiurile poligonului).

Diagrama redată în figura 3.4.10 conţine relaţii de compunere. O instanţă a clasei *Triunghi2D* este determinată de 3 obiecte de tip *Punct2D* (vârfurile triunghiului), 3 obiecte de tip *Segment2D* (laturile triunghiului) şi 3 obiecte de tip *Unghi2D* (unghiurile triunghiului). Analog o instanţă a clasei *Patrulater2D* este determinată de 4 obiecte de tip *Punct2D* (vârfurile patrulaterului), 4 obiecte de tip *Segment2D* (laturile patrulaterului) şi 4 obiecte de tip *Unghi2D* (unghiurile patrulaterului). Studiind relaţiile de compunere din figura 3.4.11 se poate observa că o instanţă a clasei *Parabola2D* este determinată de 2 obiecte de tip *Punct2D* (vârful şi focarul parabolei) şi un obiect de tip *Dreapta2D* (directoarea parabolei).

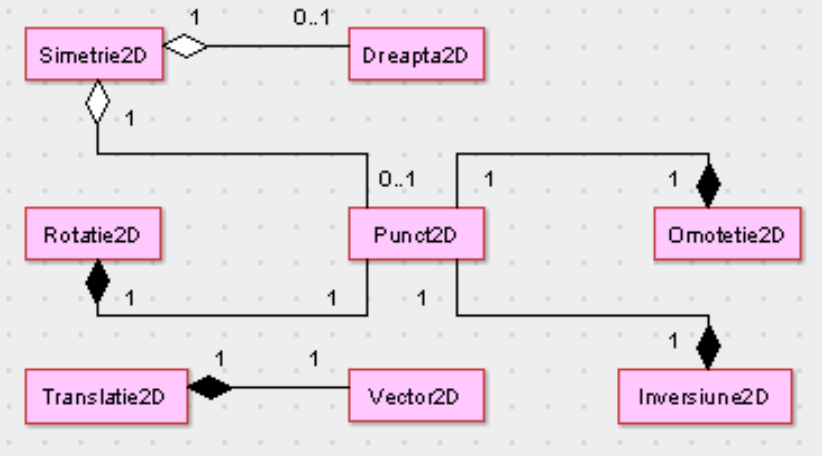


Figura 3.4.12. Diagrama ce prezintă relații de compunere şi de agregare între elementele geometrice ce caracterizează transformările geometrice în plan

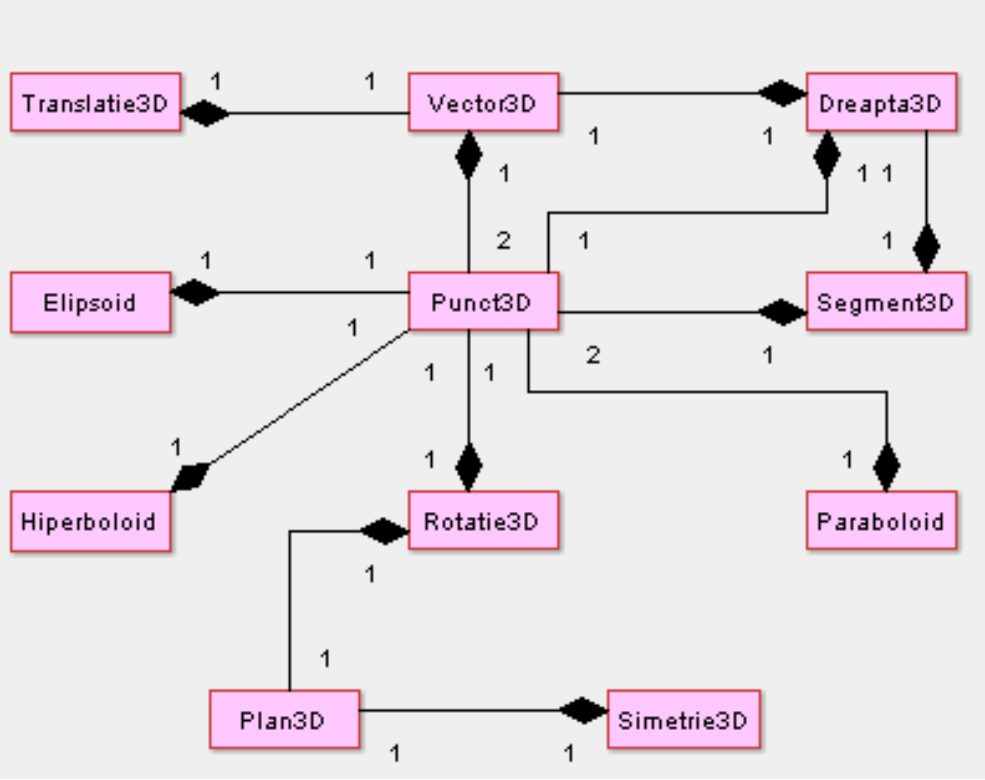


Figura 3.4.13. Diagrama ce prezintă relații de compunere între obiecte geometrice în spațiu

Diagrama redată în figura 3.4.12 conține atât relații de compunere, cât şi relații de asociere. Obiectul de tip *Punct2D* intervine în 3 relații de compunere şi o relație de asociere, astfel că poate să aparțină instanțelor de tip *Rotatie2D*, *Omotetie2D*, *Inversiune2D* şi, eventual, *Simetrie2D*.

Analizând figura 3.4.13 se poate observa că redă doar relații de compunere. O instanță a clasei *Dreapta3D* este determinată de un obiect de tip *Punct3D* şi un obiect de tip *Vector3D*, dar poate reprezenta şi o parte a unei instanțe de tip *Segment3D*. Un obiect al clasei *Plan3D* poate reprezenta atât o parte a unei instanțe de tip *Rotatie3D*, cât şi a unei instanțe de tip *Simetrie3D*.

3.4.4 Diagrama pachetelor

Instanțele claselor prezentate în anexa 1 pot fi grupate după proprietăți şi metode înrudite. Gruparea se realizează prin intermediul pachetelor [11]. În figura 3.4.14 este prezentată explicit structura pachetelor ce grupează obiecte atât din geometria plană, cât şi din geometria în spațiu. Între clasele şi instanțele claselor dintr-un pachet există relații de moştenire şi de compunere, relații ce au fost prezentate prin intermediul diagramelor din paragrafele anterioare.

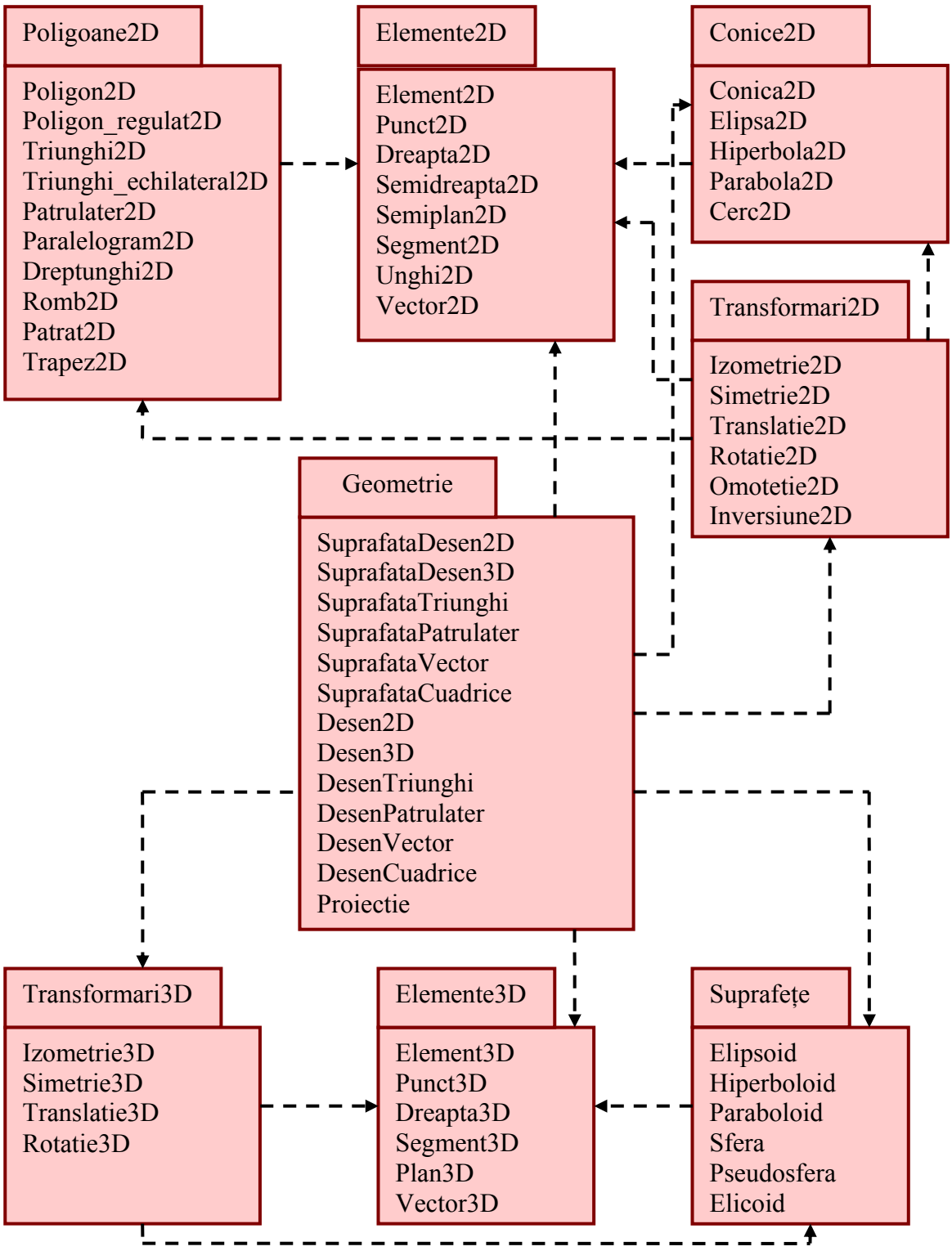


Figura 3.4.14. Diagrama pachetelor

3.4.5 Diagrame de stare

Obiectele corespunzătoare claselor prezentate au atât comportament, cât şi stare internă, cu alte cuvinte, îndeplinesc acţiuni şi deţin informaţii. Pentru a le înţelege se pot dezvolta diagramele de stare [13]. Diagramele de stare UML descriu diferitele stări în care se poate găsi un obiect şi tranziţiile dintre aceste stări [111].

În figura 3.4.15 este prezentată diagrama de stare corespunzătoare unei instanţe a clasei *Desen3D*, clasă ce implementează interfaţa *Runnable* corespunzătoare firelor de execuţie. Se observă că există şapte stări posibile pentru un obiect de acest tip: Selectare opţiuni, Selectare element geometric, Reprezentare grafică, Iniţiere animaţie, Rulare, Aşteptare şi Încheiere animaţie.

Instanţa clasei *Desen3D* se găseşte în starea *Selectare opţiuni* când s-a ales stilul de desenare a viitorului element geometric 3D, respectiv proiecţia folosită pentru reprezentarea sa, iar în starea *Selectare element geometric* când s-a creat un obiect de tip *Element3D*. Starea *Reprezentare grafică* presupune adăugarea elementului 3D la construcţia geometrică. În cazul trecerii obiectului prin celălalte stări se realizează o animaţie ce permite vizualizarea construcţiei geometrice când punctul de observare desfăşoară o rotaţie în jurul unei axe specificate.

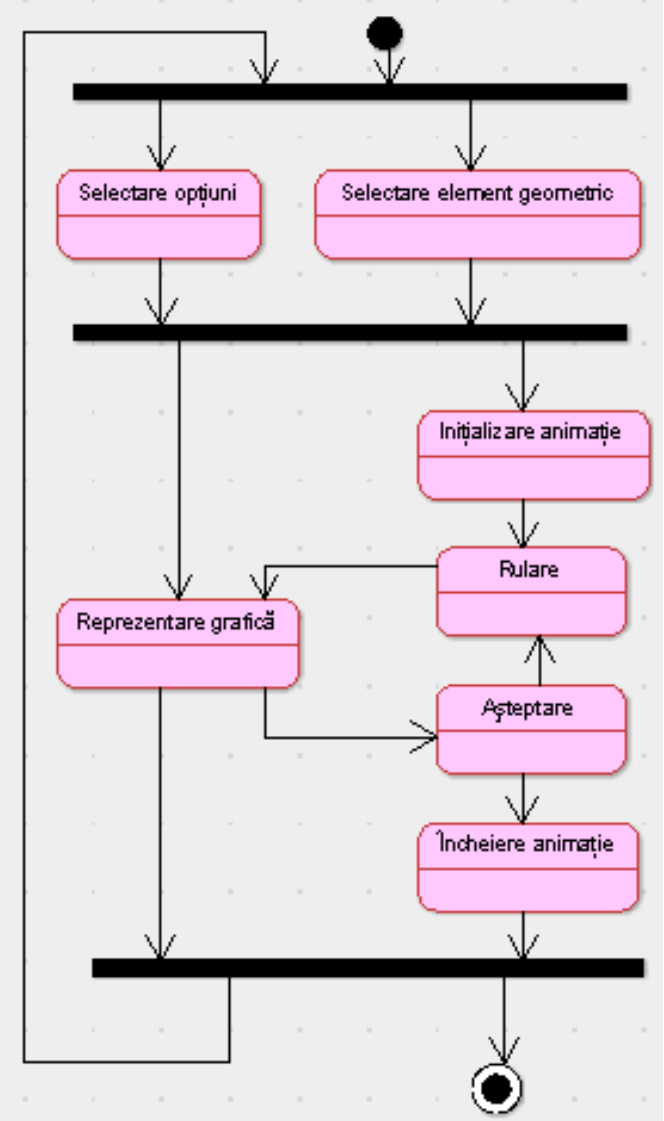


Figura 3.4.15. Diagramă de stare

3.4.6 Diagrame de secvență

Diagramele de secvență descriu comportamentul unei mulțimi de obiecte dintr-un anumit context, punând accentul pe aspectul temporal [12].

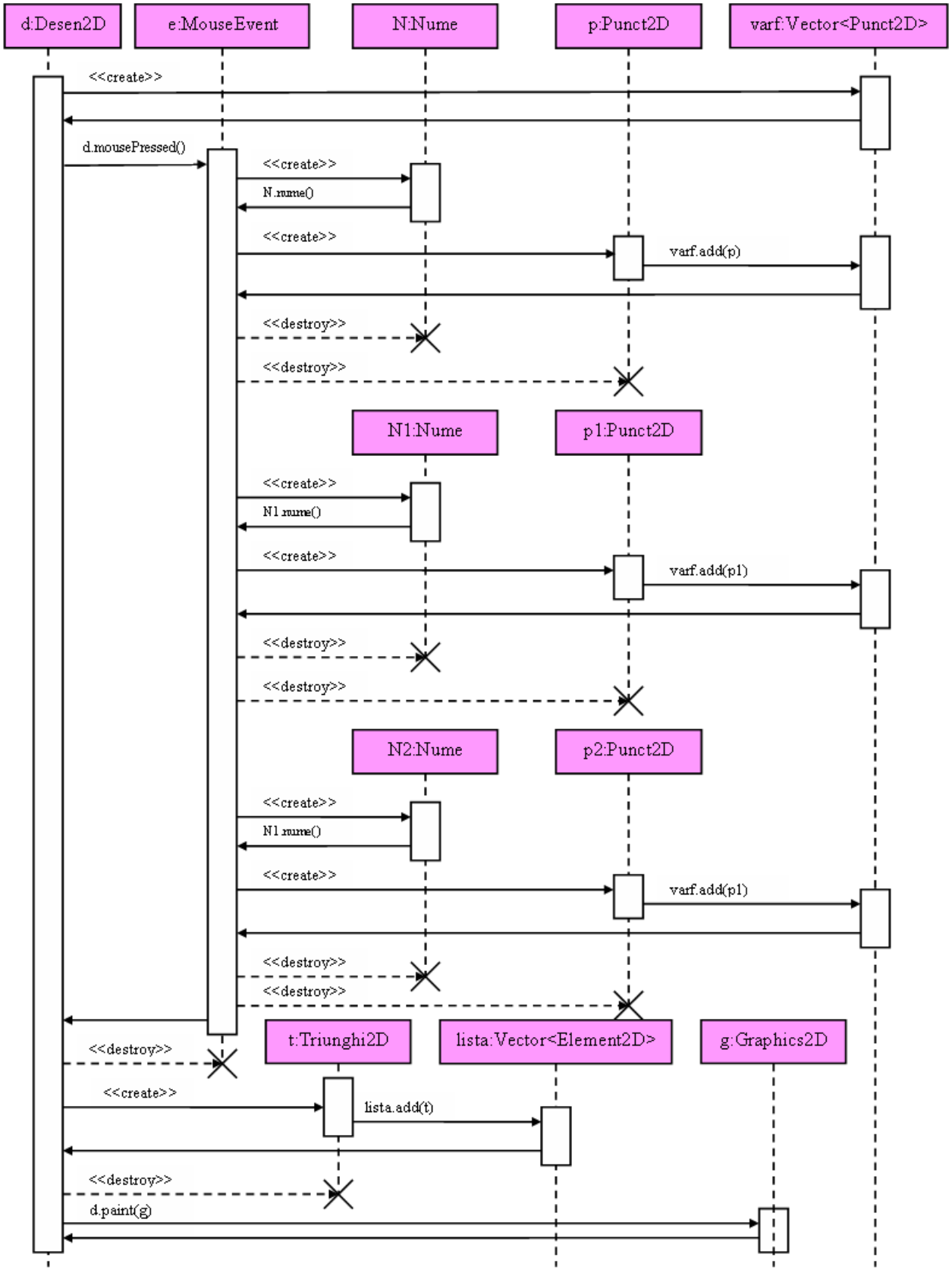


Figura 3.4.16. Diagramă de secvență pentru desenarea unui triunghi

Diagrama prezentată în figura 3.4.16 redă interacțiunile dintre obiecte care au ca scop desenarea unui triunghi ale cărui vârfuri vor fi specificate. Se observă că există interacțiuni între 12 obiecte, dintre care obiectele de tip *Desen2D*, *MouseEvent*, *Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Nume*, *Punct2D şi Vector<Punct2D>* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *Desen2D* care crează o instanță a clasei *Vector<Punct2D>*. Acum controlul este preluat de această instanță nou creată ce va permite memorarea vârfurilor triunghiului ce se va desena.

Controlul va fi redat obiectului de tip *Desen2D,* care în urma evenimentului de apăsare al butonului *mouse*-ului va trasmite controlul obiectului de tip *MouseEvent*. Se va crea o instanță a clasei *Nume*. Acum controlul este preluat de această instanță nou creată ce va permite afişarea unei ferestre în care se va introduce identificatorul punctului ce va fi instanțiat în continuare, iar apoi controlul va fi redat obiectului de tip *MouseEvent*.

Se instanțiază în continuare obiectul de tip *Punct2D*, iar apoi contrul execuției este transmis obiectului de tip *Vector<Punct2D>* pentru a adăuga punctul anterior creat în lista de vârfuri a triunghiului ce urmează a fi instanțiat.

Redându-se controlul obiectului de tip *MouseEvent*, vor fi distruse obiectele de tip *Nume* şi *Punct2D*. Se observă că linia vieții acestor două obiecte se întrerupe, prin marcare cu X, la apariția mesajului purtând marca stereotipului *<<destroy>>*.

Se repetă de două ori instanțierea de obiecte de tip *Nume* şi *Punct2D*, pentru a obține şi celelalte două vârfuri ale triunghiului, puncte ce vor fi adăugate în lista de tip *Vector<Punct2D>*.

În continuare contrul execuției este transmis obiectului de tip *Desen2D*, care va distruge instanța clasei *MouseEvent* şi va instanția obiectul de tip *Triunghi2D*. Contrul execuției este transmis obiectului de tip *Vector<Element2D>* pentru a adăuga triunghiul anterior creat în lista de elemente 2D ale construcției geometrice, iar apoi se distruge instanța clasei *Triunghi2D*.

Ultimul mesaj va duce la redesenarea construcției geometrice ce va cuprinde acum şi triunghiul anterior creat.

Diagrama prezentată în figura 3.4.17 redă interacțiunile dintre obiecte care au ca scop desenarea centrului de greutate şi a triunghiului median corespunzătoare unui triunghi oarecare. Se observă că există interacțiuni între 5 obiecte, dintre care obiectele de tip *DesenTriunghi*, *Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Element2D şi Triunghi2D* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *DesenTriunghi* care apelează obiectul de tip *Vector<Element2D>* pentru a obține centrul de greutate al triunghiului existent şi va crea o instanță a clasei *Element2D* ce va fi convertită spre tipul *Punct2D*. Acum controlul este preluat de instanța clasei *Vector<Element2D>* ce va permite adăugarea punctului în lista de elemente 2D ale construcției geometrice.

Controlul fiind redat obiectului de tip *DesenTriunghi*, se va redesena construcția geometrică ce va cuprinde acum şi centrul de greutate al triunghiului.

În continuare obiectul de tip *DesenTriunghi* va apela obiectul de tip *Vector<Element2D>* pentru a obține triunghiul median al triunghiului existent şi va crea o instanță a clasei *Triunghi2D*. Apoi controlul este preluat de instanța clasei *Vector<Element2D>* ce va permite adăugarea obiectului nou creat în lista de elemente 2D ale construcției geometrice.

Controlul va fi redat obiectului de tip *DesenTriunghi* care va distruge obiectul de tip *Triunghi2D*.

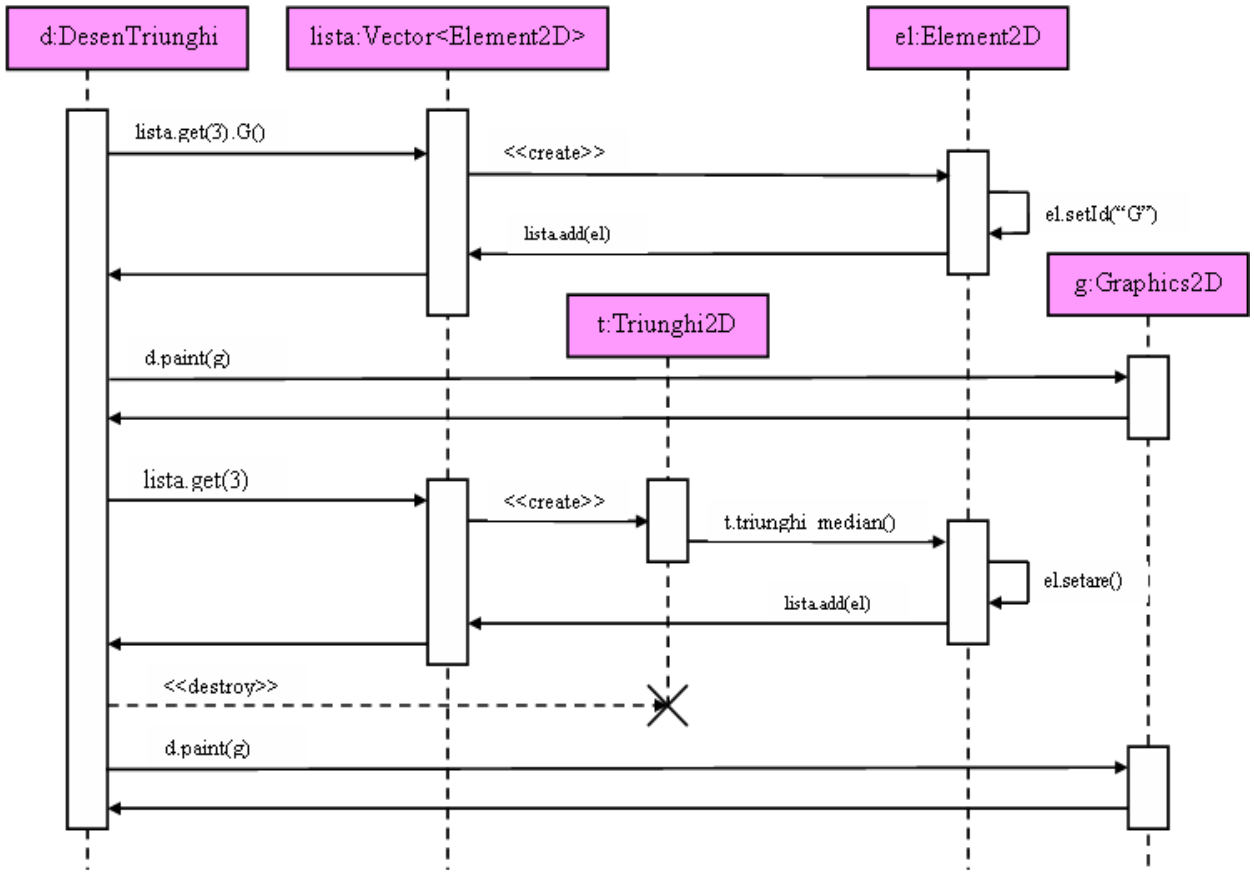


Figura 3.4.17. Diagramă de secvență pentru desenarea centrului de greutate şi a triunghiului median corespunzătoare unui triunghi oarecare

Ultimul mesaj va duce la redesenarea construcției geometrice ce va cuprinde acum şi triunghiul median anterior creat.

Diagrama prezentată în figura 3.4.18 redă interacțiunile dintre obiecte care au ca scop desenarea cercului circumscris unui triunghi oarecare, dar şi a centrului acestui cerc. Se observă că există interacțiuni între 5 obiecte, dintre care obiectele de tip *DesenTriunghi*, *Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Element2D* şi *Triunghi2D* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *DesenTriunghi* care apelează obiectul de tip *Vector<Element2D>* pentru a obține punctul de intersecție al mediatoarelor triunghiului existent şi va crea o instanță a clasei *Element2D* ce va fi convertită spre tipul *Punct2D*. Acum controlul este preluat de instanța clasei *Vector<Element2D>* ce va permite adăugarea punctului în lista de elemente 2D ale construcției geometrice.

Controlul fiind redat obiectului de tip *DesenTriunghi*, se va redesena construcția geometrică ce va cuprinde acum şi punctul de intersecție al mediatoarelor triunghiului.

În continuare obiectul de tip *DesenTriunghi* va apela obiectul de tip *Vector<Element2D>* pentru a obține triunghiul existent memorat într-o instanță a clasei *Triunghi2D*. Apoi controlul este preluat de instanța clasei *Triunghi2D* ce va permite instanțierea unui nou element geometric 2D corespunzător cercului circumscris triunghiului.

Controlul este redat obiectului de tip *Vector<Element2D>* pentru adăugarea obiectului nou creat în lista de elemente 2D ale construcției geometrice.

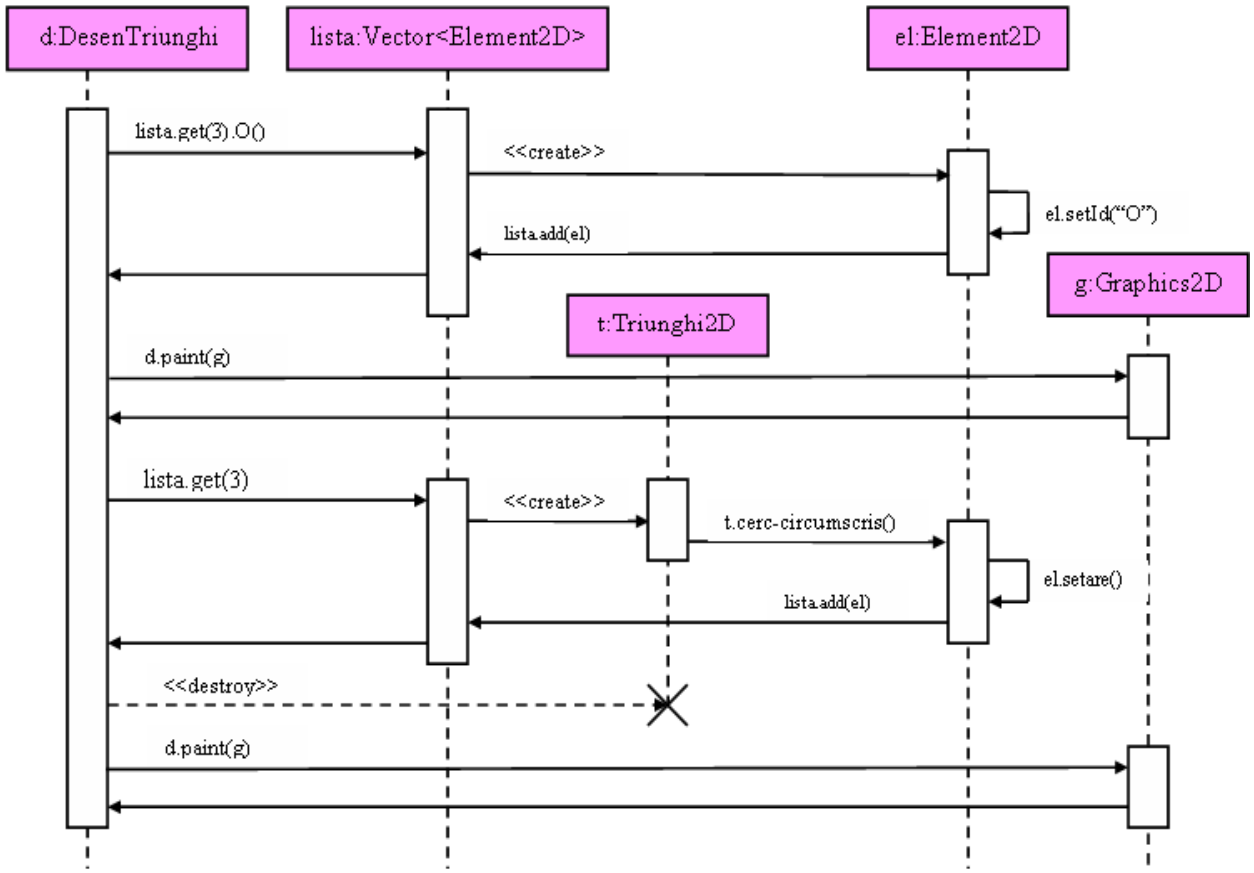


Figura 3.4.18. Diagramă de secvență pentru desenarea cercului circumscris unui triunghi oarecare, dar şi a centrului acestui cerc

Controlul va fi redat obiectului de tip *DesenTriunghi* care va distruge obiectul de tip *Triunghi2D*. Se observă că linia vieții acestui obiect se întrerupe, prin marcare cu X, la apariția mesajului purtând marca stereotipului *<<destroy>>*.

Ultimul mesaj va duce la redesenarea construcției geometrice ce va cuprinde acum şi cercul circumscris triunghiului dat.

Diagrama prezentată în figura 3.4.19 redă interacțiunile dintre obiecte care au ca scop desenarea bimedianelor unui patrulater. Se observă că există interacțiuni între 6 obiecte, dintre care obiectele de tip *DesenPatrulater*, *Vector<Element2D>, Element2D* şi *Graphics2D* sunt deja create, iar obiectele de tip *Patrulater2D şi Segment2D* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *DesenPatrulater* care apelează instanța clasei *Vector<Element2D>*.

Acum controlul fiind preluat de această instanță, se crea o instanță a clasei *Patrulater2D* corespunzătoare patrulaterului pentru care se vor desena bimedianele sale. Se determină o primă bimediană a patrulaterului prin instanțierea unui obiect de tip *Segment2D*, iar apoi contrul execuției este transmis obiectului de tip *Vector<Element2D>* pentru a adăuga segmentul anterior creat în lista de elemente 2D ale construcției geometrice.

Pentru segmentul anterior creat se vor determina cele două extremități ale sale şi vor fi adăugate în lista de elemente geometrice. Se repetă acest proces de două ori pentru a obține şi celelalte două bimediane ale patrulaterului, bimediane ce vor fi adăugate în lista de tip *Vector<Element2D>* împreună cu extremitățile lor.

În continuare contrul execuției este transmis obiectului de tip *DesenPatrulater*, care va distruge instanțele claselor *Patrulater2D* şi *Segment2D*. Se observă că linia vieții acestor obiecte se întrerupe, prin marcare cu X, la apariția mesajului purtând marca stereotipului <<*destroy*>>.

Ultimul mesaj va duce la redesenarea construcției geometrice ce va cuprinde acum şi bimedianele patrulaterului dat.

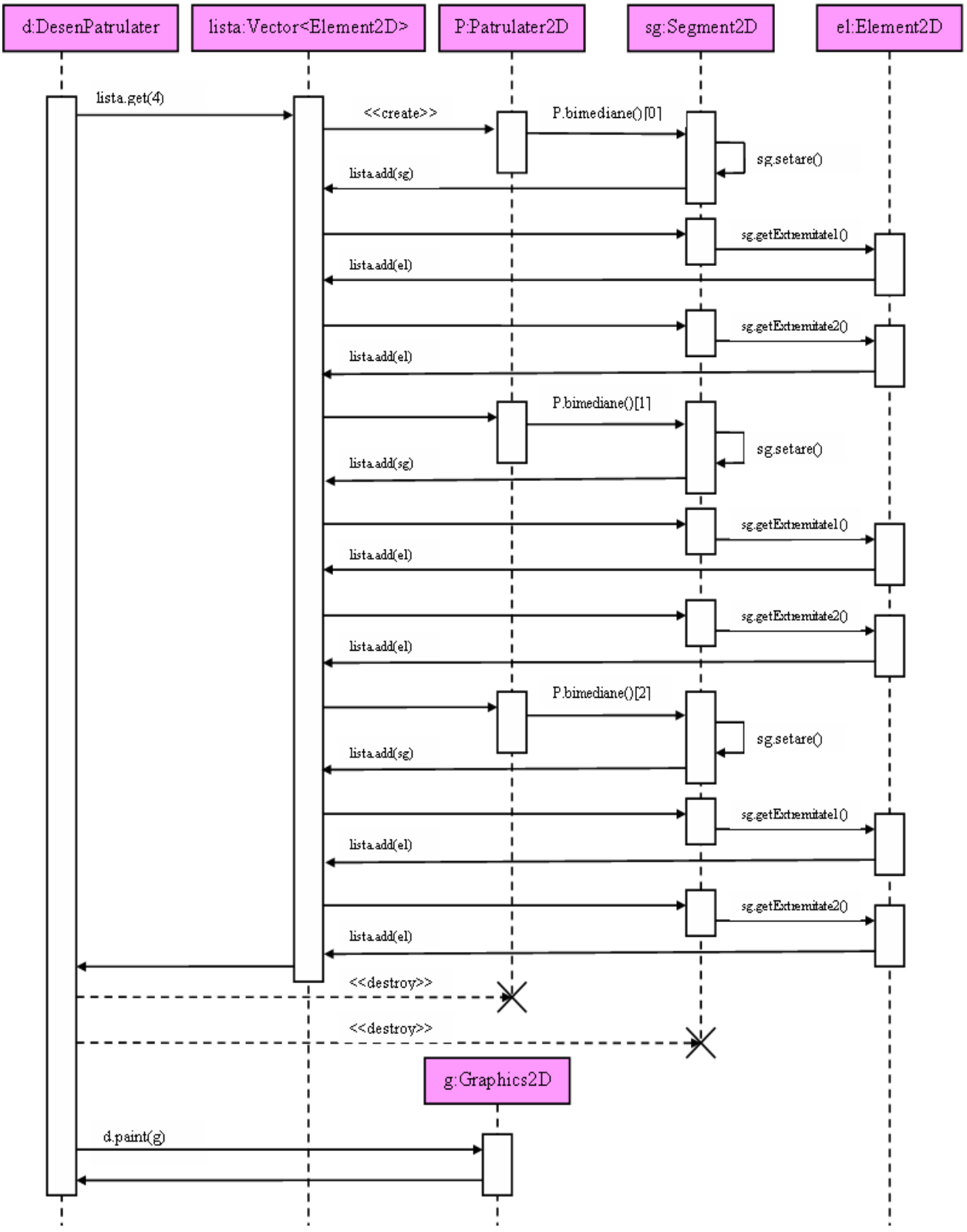


Figura 3.4.19. Diagramă de secvență pentru desenarea bimedianelor unui patrulater

Diagrama prezentată în figura 3.4.20 redă interacțiunile dintre obiecte care au ca scop desenarea elipsei determinată de focare şi un parametru. Se observă că există interacțiuni între 11 obiecte, dintre care obiectele de tip *Desen2D*, *Vector<Punct2D>, Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Parametru*, *MouseEvent*, *Punct2D, Element2D şi Elipsa2D* se vor instanția pe parcursul interacțiunilor.

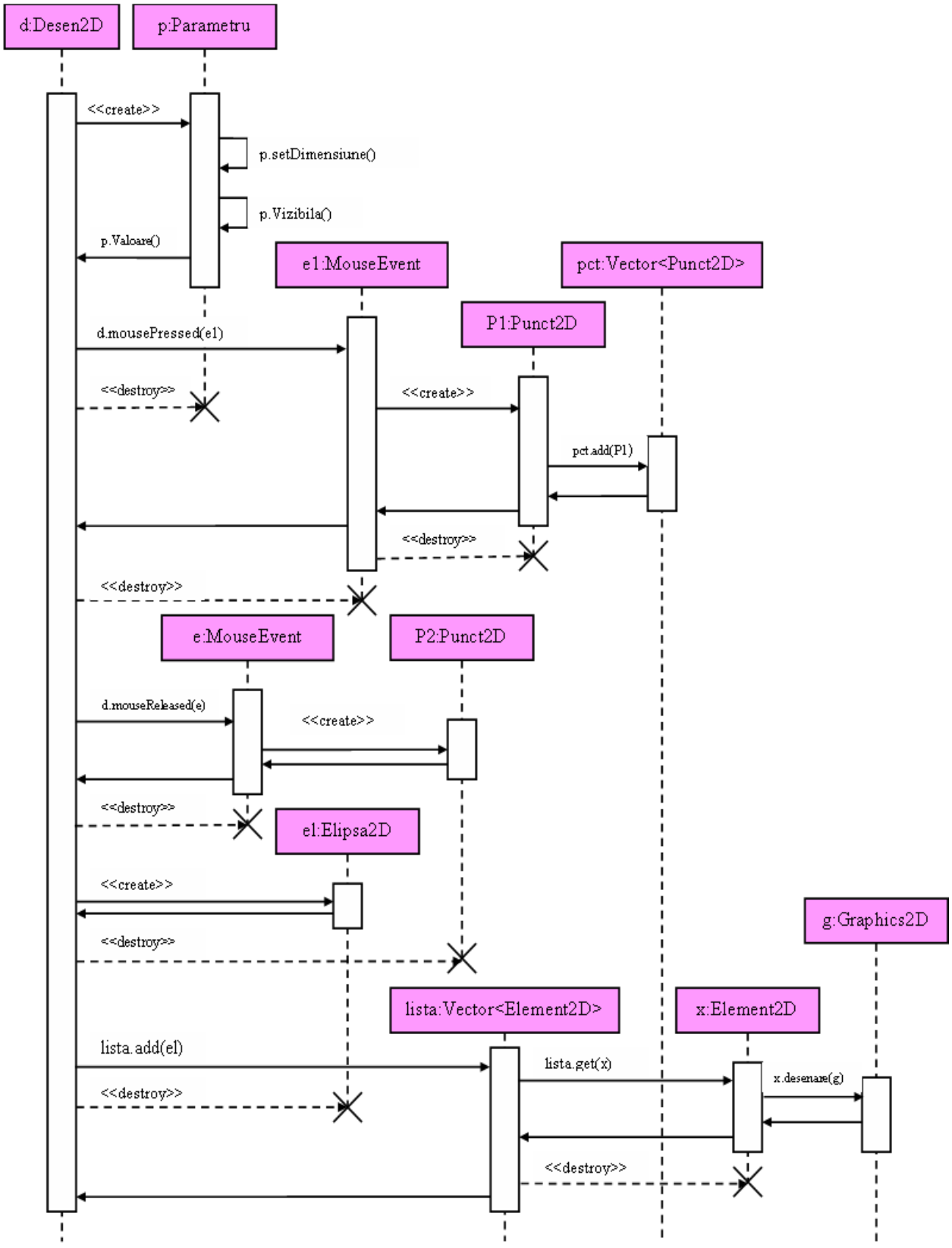


Figura 3.4.20. Diagramă de secvență pentru desenarea elipsei determinată de focare şi un parametru

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuţiei este preluat de obiectul de tip *Desen2D* care crează o instanţă a clasei *Parametru*. Acum controlul este preluat de această instanţă nou creată ce va permite afişarea unei ferestre în care se va introduce un parametru al elipsei.

Aceste valori vor fi returnate, iar controlul va fi redat obiectului de tip *Desen2D* care va instanţia în continuare obiectul de tip *MouseEvent*, iar apoi va distruge obiectul de tip *Parametru*. Se observă că linia vieţii acestui obiect se întrerupe, prin marcare cu X, la apariţia mesajului purtând marca stereotipului <<*destroy*>>.

În urma evenimentului de apăsare a butonului *mouse*-ului se instanţiază primul obiect de tip *Punct2D*, iar apoi contrul execuţiei este transmis obiectului de tip *Vector<Punct2D>* pentru a adăuga punctul anterior creat în lista de puncte folosită pentru memorarea focarelor elipsei. Controlul este redat obiectului de tip *Desen2D*, care va distruge obiectele de tip *MouseEvent* şi *Punct2D*.

În continuare are loc instanţierea unui nou obiect de tip *MouseEvent*. În urma evenimentului de eliberare a butonului *mouse*-ului se instanţiază al doilea obiect de tip *Punct2D*.

Controlul este redat obiectului de tip *Desen2D*, care va distruge obiectul de tip *MouseEvent* şi va instanţia în continuare obiectul de tip *Elipsa2D*, iar apoi va distruge obiectul de tip *Punct2D*.

În continuare contrul execuţiei este transmis obiectului de tip *Vector<Element2D>* pentru a adăuga elipsa anterior creată în lista de elemente 2D ale construcţiei geometrice, iar apoi se distruge instanţa clasei *Elipsa2D*.

Ultimul mesaj va duce la redesenarea construcţiei geometrice ce va cuprinde acum şi elipsa anterior creată, prin utilizarea obiectului de tip *Graphics2D*.

Diagrama prezentată în figura 3.4.21 redă interacţiunile dintre obiecte care au ca scop desenarea hiperbolei determinată de focare şi un parametru. Se observă că există interacţiuni între 11 obiecte, dintre care obiectele de tip *Desen2D*, *Vector<Punct2D>, Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Parametru*, *MouseEvent*, *Punct2D, Element2D şi Hiperbola2D* se vor instanţia pe parcursul interacţiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuţiei este preluat de obiectul de tip *Desen2D* care crează o instanţă a clasei *Parametru*. Acum controlul este preluat de această instanţă nou creată ce va permite afişarea unei ferestre în care se va introduce un parametru al hiperbolei.

Aceste valori vor fi returnate, iar controlul va fi redat obiectului de tip *Desen2D* care va instanţia în continuare obiectul de tip *MouseEvent*, iar apoi va distruge obiectul de tip *Parametru*. Se observă că linia vieţii acestui obiect se întrerupe, prin marcare cu X, la apariţia mesajului purtând marca stereotipului <<*destroy*>>.

În urma evenimentului de apăsare a butonului *mouse*-ului se instanţiază primul obiect de tip *Punct2D*, iar apoi contrul execuţiei este transmis obiectului de tip *Vector<Punct2D>* pentru a adăuga punctul anterior creat în lista de puncte folosită pentru memorarea focarelor hiperbolei. Controlul este redat obiectului de tip *Desen2D*, care va distruge obiectele de tip *MouseEvent* şi *Punct2D*.

În continuare are loc instanţierea unui nou obiect de tip *MouseEvent*. În urma evenimentului de eliberare a butonului *mouse*-ului se instanţiază al doilea obiect de tip *Punct2D*.

Controlul este redat obiectului de tip *Desen2D*, care va distruge obiectul de tip *MouseEvent* şi va instanţia în continuare obiectul de tip *Hiperbola2D*, iar apoi va distruge obiectul de tip *Punct2D*. Se observă că linia vieţii acestui obiect se întrerupe, prin marcare cu X, la apariţia mesajului purtând marca stereotipului <<*destroy*>>.

În continuare contrul execuției este transmis obiectului de tip *Vector<Element2D>* pentru a adăuga hiperbola anterior creată în lista de elemente 2D ale construcției geometrice, iar apoi se distruge instanța clasei *Hiperbola2D*.

Ultimul mesaj va duce la redesenarea construcției geometrice ce va cuprinde acum şi hiperbola anterior creată, prin utilizarea obiectului de tip *Graphics2D*.

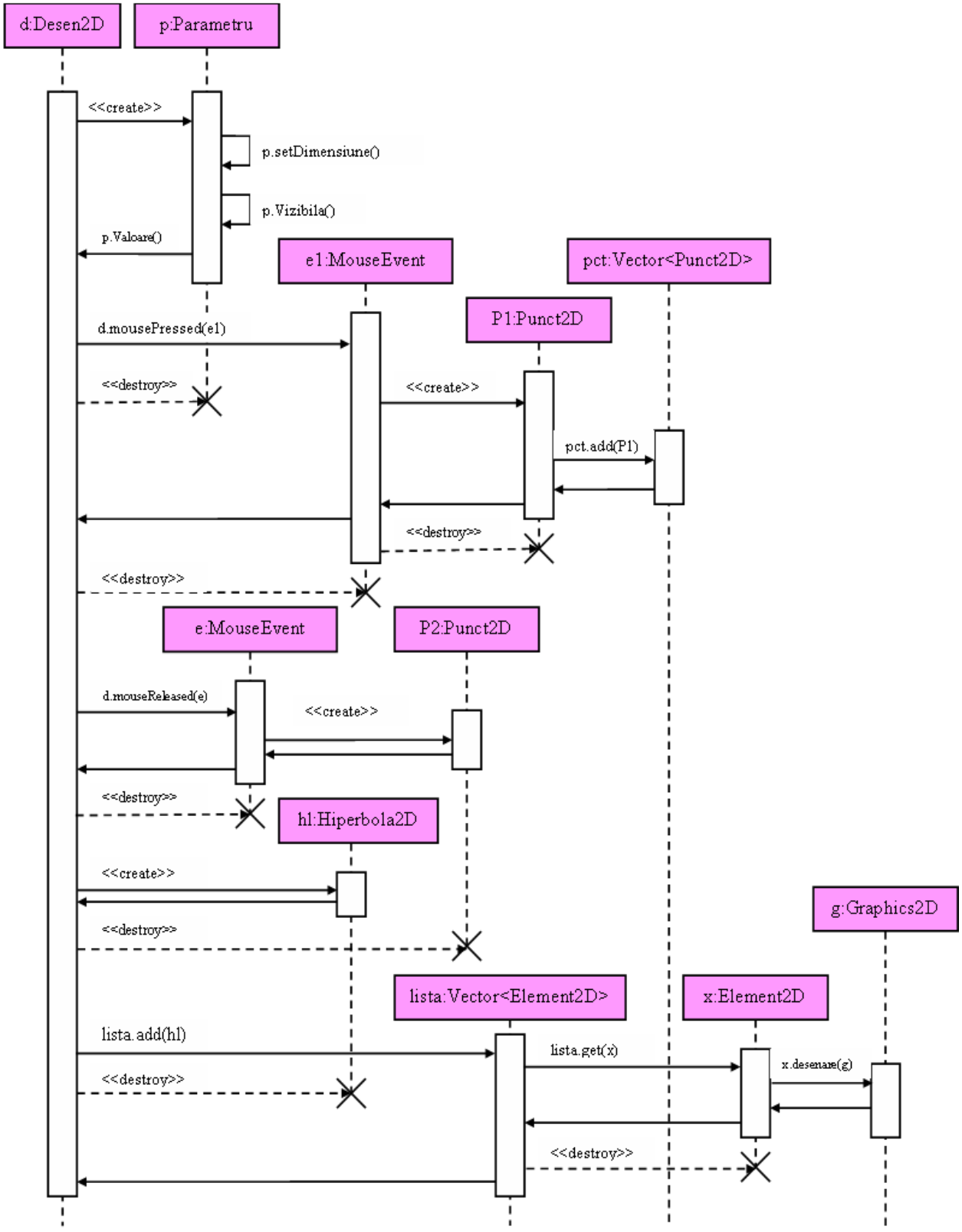


Figura 3.4.21. Diagramă de secvență pentru desenarea hiperbolei determinată de focare şi un parametru

Diagrama prezentată în figura 3.4.22 redă interacțiunile dintre obiecte care au ca scop desenarea parabolei determinată de focar şi de dreapta directoare. Se observă că există interacțiuni între 9 obiecte, dintre care obiectele de tip *Desen2D*, *Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *MouseEvent*, *Punct2D, Dreapta2D, Element2D şi Parabola2D* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *Desen2D* care crează o instanță a clasei *MouseEvent*. În urma evenimentului de eliberare a butonului *mouse*-ului se instanțiază obiect de tip *Punct2D*.

Controlul este redat obiectului de tip *Desen2D*, care va distruge obiectul de tip *MouseEvent* şi va apela în continuare obiectul de tip *Vector<Element2D>*, care va duce la instanțierea unui obiect de tip *Dreapta2D* ce va reprezenta directoarea parabolei ce se va desena.

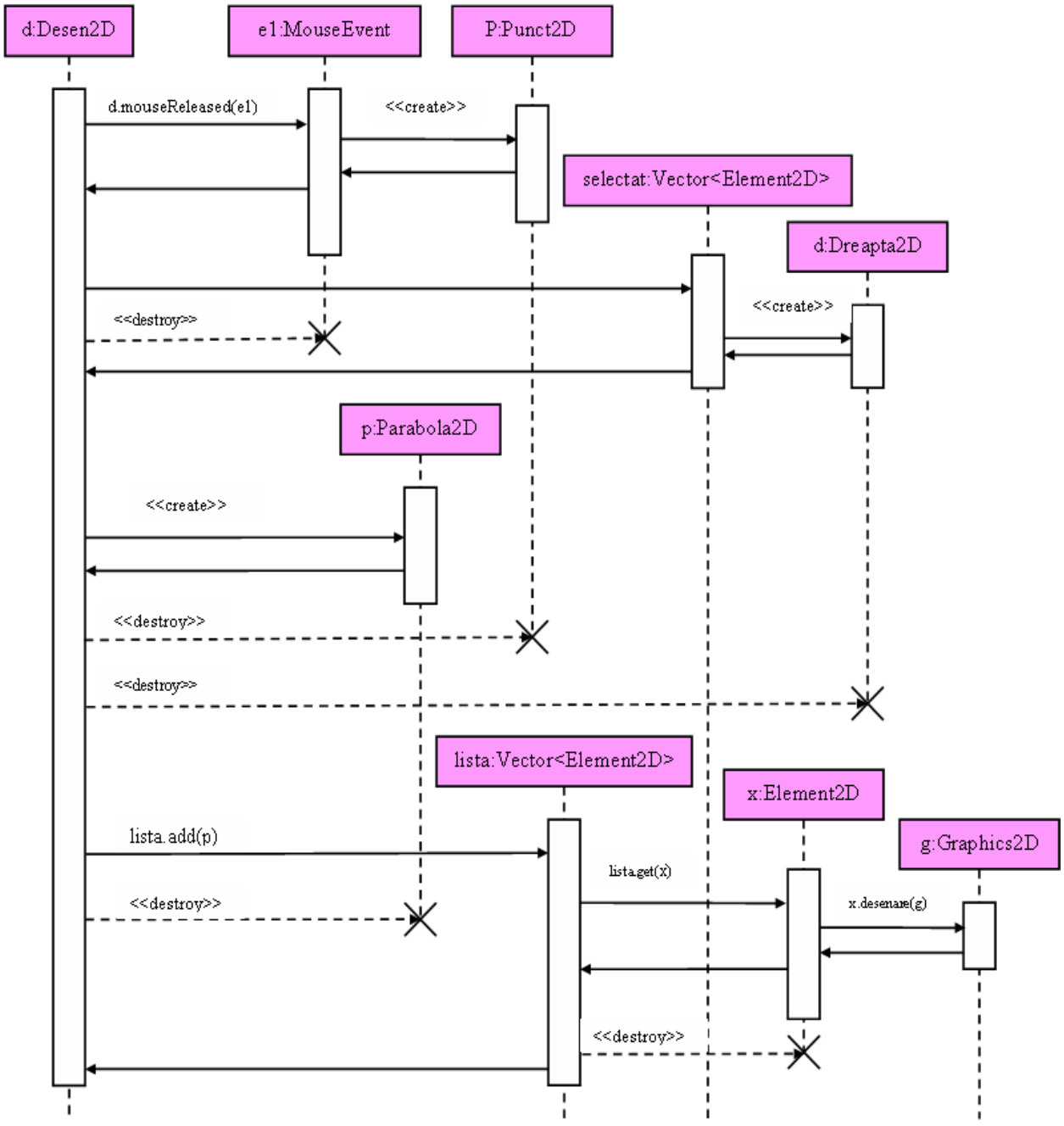


Figura 3.4.22. Diagramă de secvență pentru desenarea parabolei determinată de focar şi dreapta directoare

Redându-se obiectului de tip *Desen2D* controlul, va fi instanțiat în continuare obiectul de tip *Parabola2D*, iar apoi se vor distruge obiectele de tip *Punct2D* şi *Dreapta2D*.

În continuare contrul execuției este transmis obiectului de tip *Vector<Element2D>* pentru a adăuga parabola anterior creată în lista de elemente 2D ale construcției geometrice, iar apoi se distruge instanța clasei *Parabola2D*.

Ultimul mesaj va duce la redesenarea construcției geometrice ce va cuprinde acum şi parabola anterior creată, prin utilizarea obiectului de tip *Graphics2D*.

Diagrama prezentată în figura 3.4.23 redă interacțiunile dintre obiecte care au ca scop desenarea tangentei la elipsă într-un punct dat. Se observă că există interacțiuni între 9 obiecte, dintre care obiectele de tip *Desen2D*, *Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *MouseEvent*, *Punct2D, Dreapta2D, Element2D şi Elipsa2D* se vor instanția pe parcursul interacțiunilor.

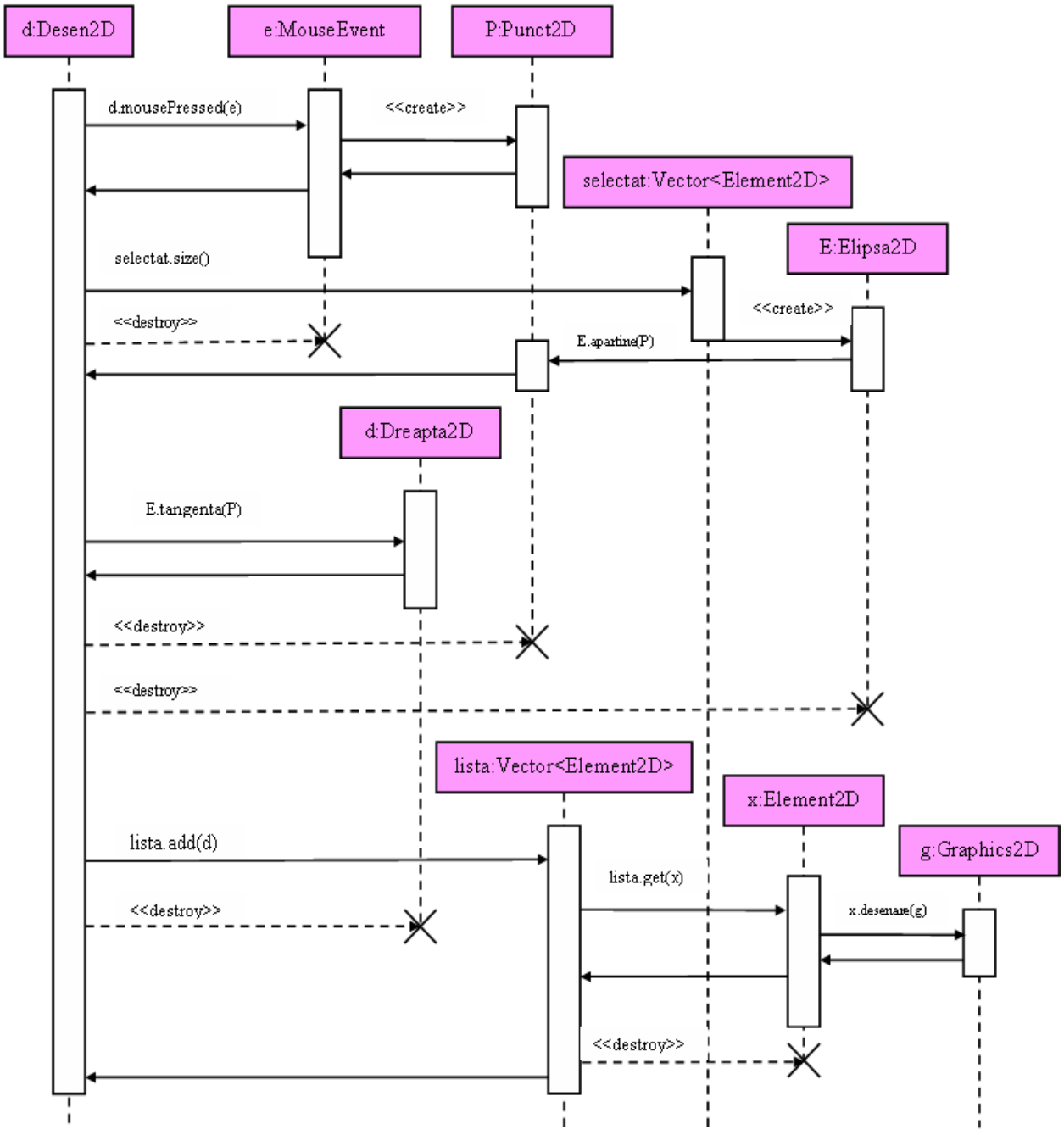


Figura 3.4.23. Diagramă de secvență pentru desenarea tangentei într-un punct la elipsă

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *Desen2D* care crează o instanță a clasei *MouseEvent*.

În urma evenimentului de apăsare a butonului *mouse*-ului se instanțiază obiect de tip *Punct2D*. Controlul este redat obiectului de tip *Desen2D*, care va distruge obiectul de tip *MouseEvent* şi va apela în continuare obiectul de tip *Vector<Element2D>*, care va duce la instanțierea unui obiect de tip *Elipsa2D* reprezentând elipsa a cărei tangentă se va desena.

În acest moment controlul este transmis obiectului de tip *Punct2D* pentru a verifica dacă acesta aparține elipsei.

Redându-se obiectului de tip *Desen2D* controlul, va fi instanțiat în continuare obiectul de tip *Dreapta2D*, reprezentând tangenta la elipsă, iar apoi se vor distruge obiectele de tip *Punct2D* şi *Elipsa2D*.

În continuare contrul execuției este transmis obiectului de tip *Vector<Element2D>* pentru a adăuga dreapta tangentă anterior creată în lista de elemente 2D ale construcției geometrice, iar apoi se distruge instanța clasei *Dreapta2D*.

Obiectul de tip *Vector<Element2D>* având controlul în acest moment, va instanția obiecte de tip *Element2D* pentru care se va apela metoda de desenare a acestora prin transmiterea repetată a controlului între obiectele de tip *Element2D* şi *Graphics2D*.

Astfel se va redesena construcția geometrică ce va cuprinde acum şi dreapta tangentă la elipsă anterior creată, prin utilizarea obiectului de tip *Graphics2D*.

Diagrama prezentată în figura 3.4.24 redă interacțiunile dintre obiecte care au ca scop desenarea normalei la hiperbolă într-un punct dat. Se observă că există interacțiuni între 9 obiecte, dintre care obiectele de tip *Desen2D*, *Vector<Element2D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *MouseEvent*, *Punct2D, Dreapta2D, Element2D şi Hiperbola2D* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *Desen2D* care crează o instanță a clasei *MouseEvent*.

În urma evenimentului de apăsare a butonului *mouse*-ului se instanțiază obiect de tip *Punct2D*. Controlul este redat obiectului de tip *Desen2D*, care va distruge obiectul de tip *MouseEvent* şi va apela în continuare obiectul de tip *Vector<Element2D>*, care va duce la instanțierea unui obiect de tip *Hiperbola2D* reprezentând hiperbola a cărei normală se va desena.

În acest moment controlul este transmis obiectului de tip *Punct2D* pentru a verifica dacă acesta aparține hiperbolei.

Redându-se obiectului de tip *Desen2D* controlul, va fi instanțiat în continuare obiectul de tip *Dreapta2D*, reprezentând normala la hiperbolă, iar apoi se vor distruge obiectele de tip *Punct2D* şi *Hiperbola2D*.

În continuare contrul execuției este transmis obiectului de tip *Vector<Element2D>* pentru a adăuga dreapta normală anterior creată în lista de elemente 2D ale construcției geometrice, iar apoi se distruge instanța clasei *Dreapta2D*.

Obiectul de tip *Vector<Element2D>* având controlul în acest moment, va instanția obiecte de tip *Element2D* pentru care se va apela metoda de desenare a acestora prin transmiterea repetată a controlului între obiectele de tip *Element2D* şi *Graphics2D*.

Astfel se va redesena construcția geometrică ce va cuprinde acum şi dreapta normală la hiperbolă anterior creată, prin utilizarea obiectului de tip *Graphics2D*.

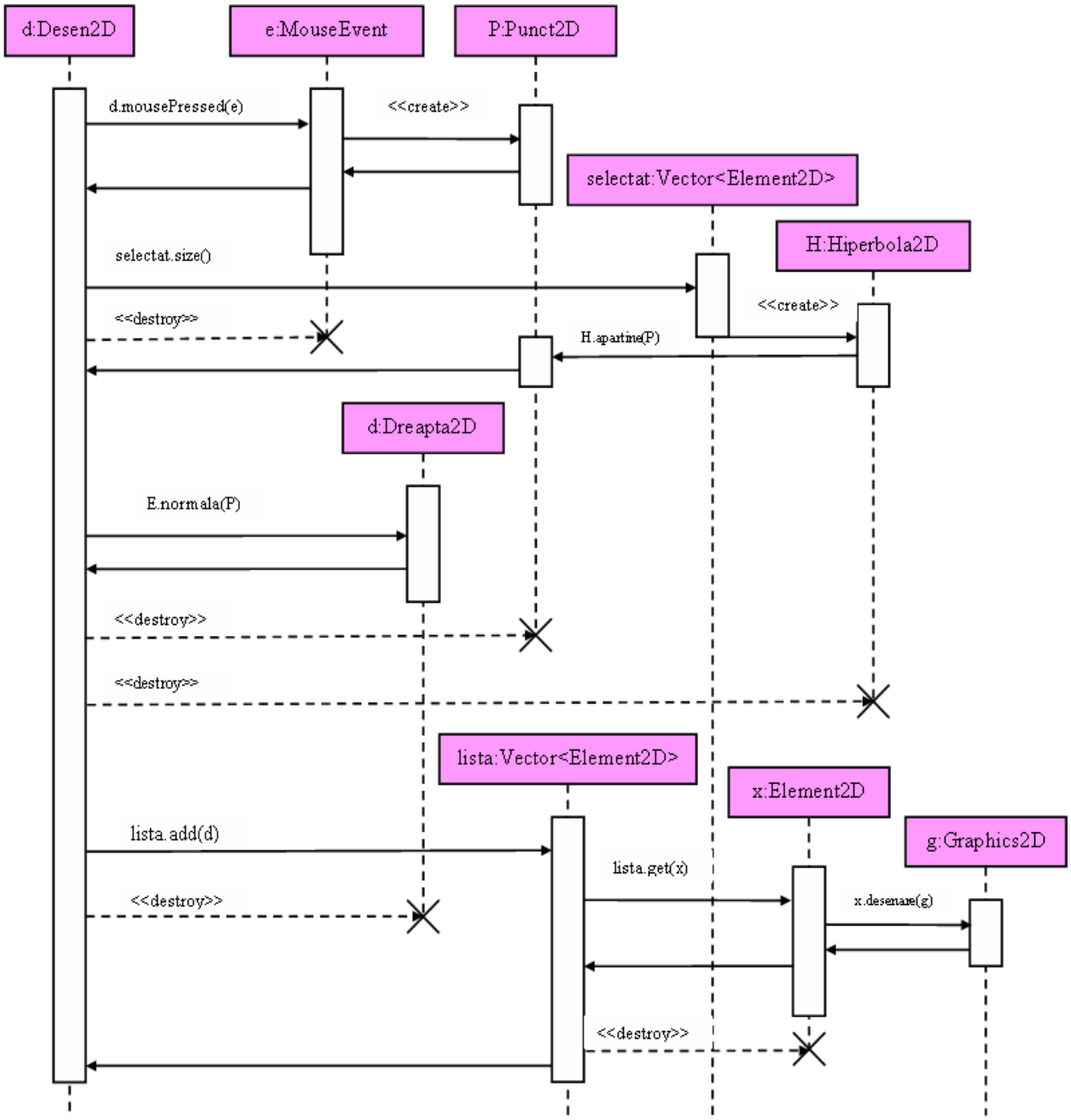


Figura 3.4.24. Diagramă de secvență pentru desenarea normalei într-un punct la hiperbolă

Diagrama prezentată în figura 3.4.25 redă interacțiunile dintre obiecte care au ca scop desenarea unui elipsoid al cărui centru este originea sistemului cartezian de axe. Se observă că există interacțiuni între cinci obiecte, dintre care obiectele de tip *Desen3D*, *Vector<Element3D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Parametru3* şi *Elipsoid* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *Desen3D* care crează o instanță a clasei *Parametru3*. Acum controlul este preluat de această instanță nou creată ce va permite afişarea unei ferestre în care se vor introduce parametrii elipsoidului.

Aceste valori vor fi returnate, iar controlul va fi redat obiectului de tip *Desen3D* care va instanția în continuare obiectul de tip *Elipsoid*, iar apoi va distruge obiectul de tip *Parametru3*. Se observă că linia vieții acestui obiect se întrerupe, prin marcare cu X, la apariția mesajului purtând marca stereotipului *<<destroy>>*.

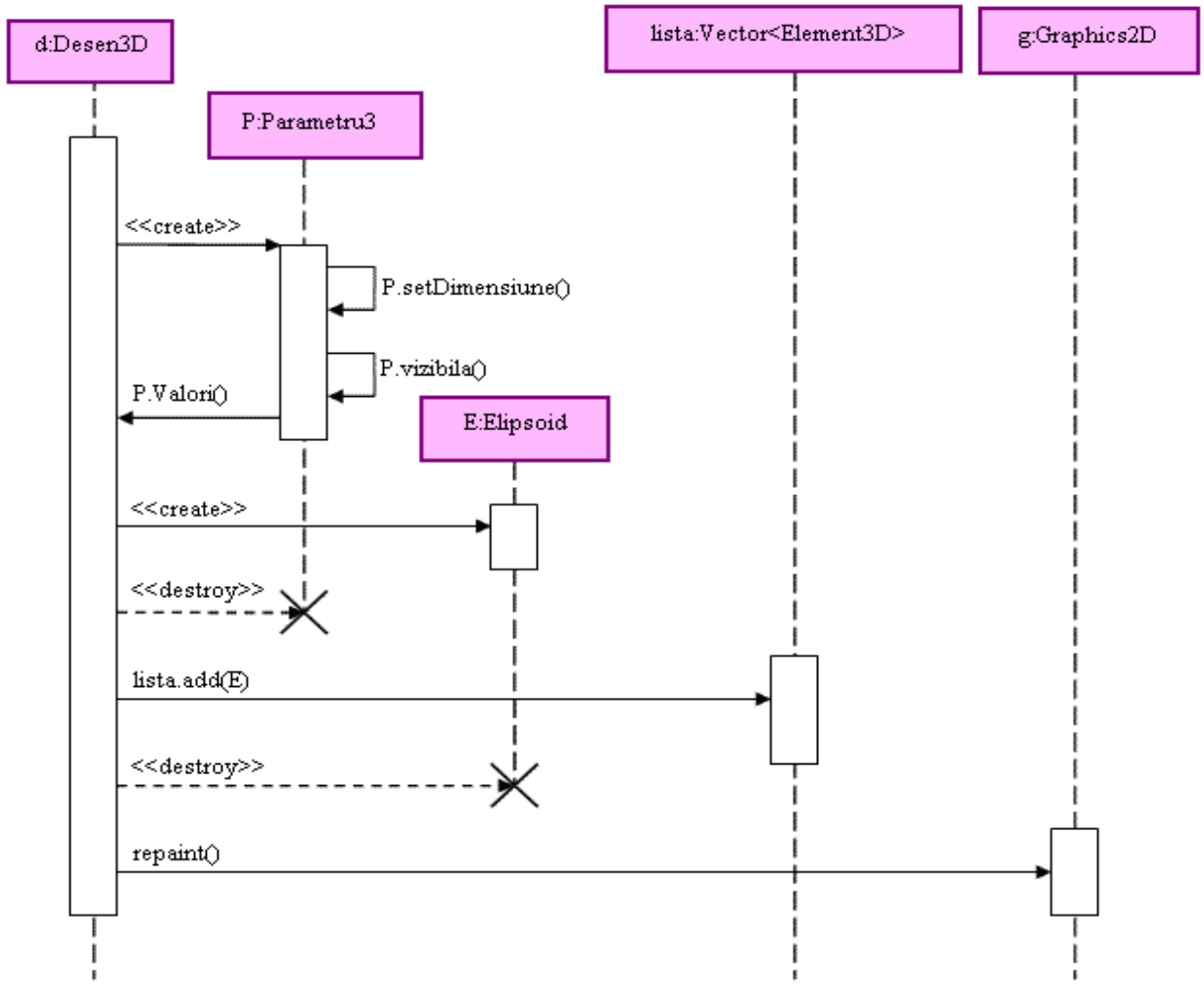


Figura 3.4.25. Diagramă de secvență pentru desenarea elipsoidului determinat de cei trei parametri

În continuare contrul execuției este transmis obiectului de tip *Vector<Element3D>* pentru a adăuga elipsoidul anterior creat în lista de elemente 3D ale construcției geometrice, iar apoi se distruge instanța clasei *Elipsoid*. Ultimul mesaj va duce la redesenarea construcției geometrice ce va cuprinde acum şi elipsoidul anterior creat.

Diagrama prezentată în figura 3.4.26 redă interacțiunile dintre obiecte care au ca scop desenarea planului tangent la hiperboloid într-un punct. Se observă că există interacțiuni între nouă obiecte, dintre care obiectele de tip *Desen3D*, *Vector<Element3D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Parametru2, Punct3D, Plan3D, Element3D* şi *Hiperboloid* se vor instanția pe parcursul interacțiunilor.

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *Desen3D* care crează o instanță a clasei *Parametru2*. Acum controlul este preluat de această instanță nou creată ce va permite afişarea unei ferestre în care se vor introduce date pentru determinarea unui punct al hiperboloidului.

Redându-se obiectului de tip *Desen3D* controlul, va fi instanțiat în continuare obiectul de tip *Punct3D*, iar apoi se va distruge obiectul de tip *Parametru2*. În continuare controlul execuției este transmis obiectului de tip *Vector<Element3D>* care va duce la instanțierea unui obiect de tip *Hiperboloid* reprezentând hiperboloidul a cărui plan tangent se va desena. Va fi instanțiat în continuare obiectul de tip *Plan3D*, reprezentând planul tangent la hiperboloid, iar apoi se vor distruge obiectele de tip *Punct3D* şi *Hiperboloid*.

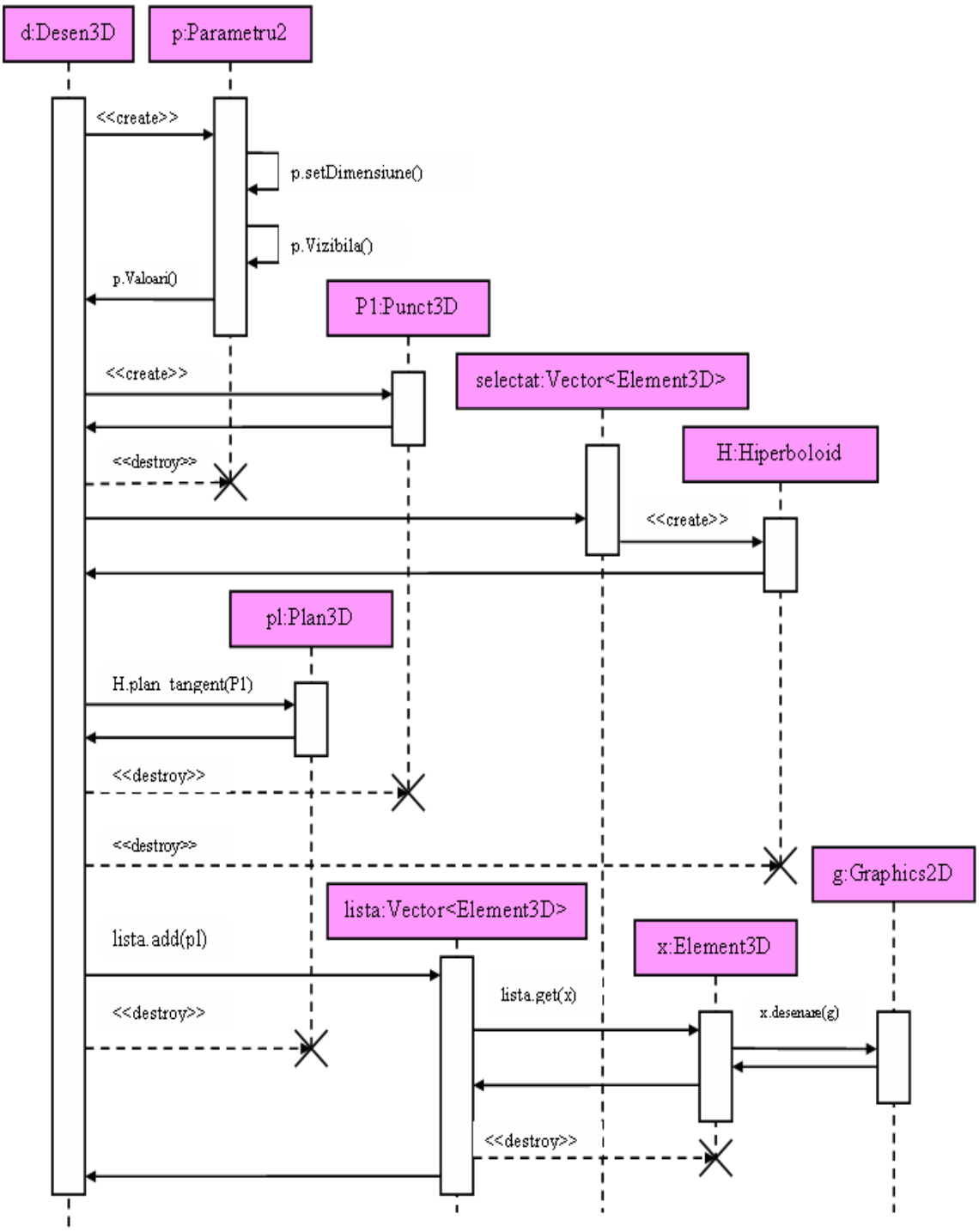


Figura 3.4.26. Diagramă de secvență pentru desenarea planului tangent la hiperboloid într-un punct

În continuare contrul execuției este transmis obiectului de tip *Vector<Element3D>* pentru a adăuga planul tangent anterior creat în lista de elemente 3D ale construcției geometrice, iar apoi se distruge instanța clasei *Plan3D*. În final se va redesena construcția geometrică ce va cuprinde acum şi planul tangent la hiperboloid anterior creat, prin utilizarea obiectului de tip *Graphics2D*.

Diagrama prezentată în figura 3.4.27 redă interacțiunile dintre obiecte care au ca scop desenarea normalei la elipsoid într-un punct. Se observă că există interacțiuni între nouă obiecte, dintre care obiectele de tip *Desen3D*, *Vector<Element3D>* şi *Graphics2D* sunt deja create, iar obiectele de tip *Parametru2, Punct3D, Dreapta3D, Element3D* şi *Elipsoid* se vor instanția pe parcursul interacțiunilor.

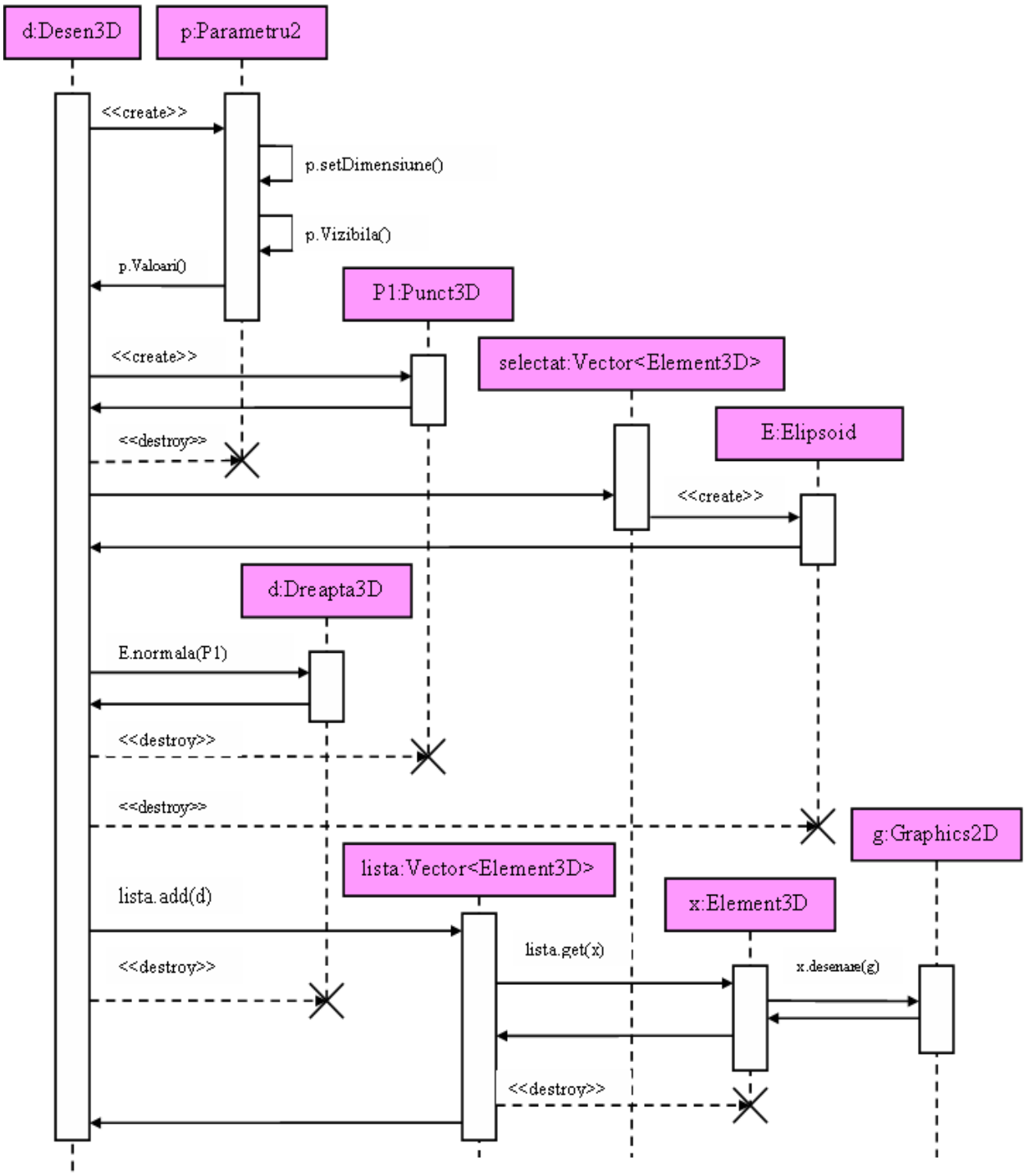


Figura 3.4.27. Diagramă de secvență pentru desenarea normalei la elipsoid într-un punct

Aceste obiecte sunt reprezentate pe axa Ox, iar pe axa Oy sunt reprezentate mesajele ordonate crescător în timp. La început controlul execuției este preluat de obiectul de tip *Desen3D* care crează o instanță a clasei *Parametru2*. Acum controlul este preluat de această instanță nou creată ce va permite afişarea unei ferestre în care se vor introduce date pentru determinarea unui punct al elipsoidului.

Redându-se obiectului de tip *Desen3D* controlul, va fi instanțiat în continuare obiectul de tip *Punct3D*, iar apoi se va distruge obiectul de tip *Parametru2*. În continuare controlul execuției este transmis obiectului de tip *Vector<Element3D>* care va duce la instanțierea unui obiect de tip *Elipsoid* reprezentând elipsoidul a cărei normală se va desena. Va fi instanțiat în continuare obiectul de tip *Dreapta3D*, reprezentând normala la elipsoid, iar apoi se vor distruge obiectele de tip *Punct3D* şi *Elipsoid*.

În continuare controlul execuției este transmis obiectului de tip *Vector<Element3D>* pentru a adăuga normalei anterior create în lista de elemente 3D ale construcției geometrice, iar apoi se distruge instanța clasei *Dreapta3D*. În final se va redesena construcția geometrică ce va cuprinde acum şi normala la elipsoid anterior creată, prin utilizarea obiectului de tip *Graphics2D*.

3.4.7 Diagrame de colaborare

Diagramele de colaborare descriu, ca şi diagramele de secvență, comportamentul unei mulțimi de obiecte dintr-un anumit context, punând accentul pe organizarea obiectelor care participă la o interacțiune [9]. Aceste diagrame sunt grafuri care au în calitate de vârfuri obiecte ce participă la interacțiune, iar arcele reprezintă legăturile dintre instanțe. În următoarele 8 diagrame sunt prezentate exemple de astfel de diagrame.

Diagrama prezentată în figura 3.4.28 redă interacțiunile dintre obiecte care compun interfața principală. Se observă că există interacțiuni între obiecte de tip: *Geometrie*, *ActionEvent*, *JButton*, *SuprafataDesen2D, SuprafataDesen3D, SuprafataTriunghi, SuprafataPatrulater, SuprafataVector* şi *SuprafataCuadrice*.

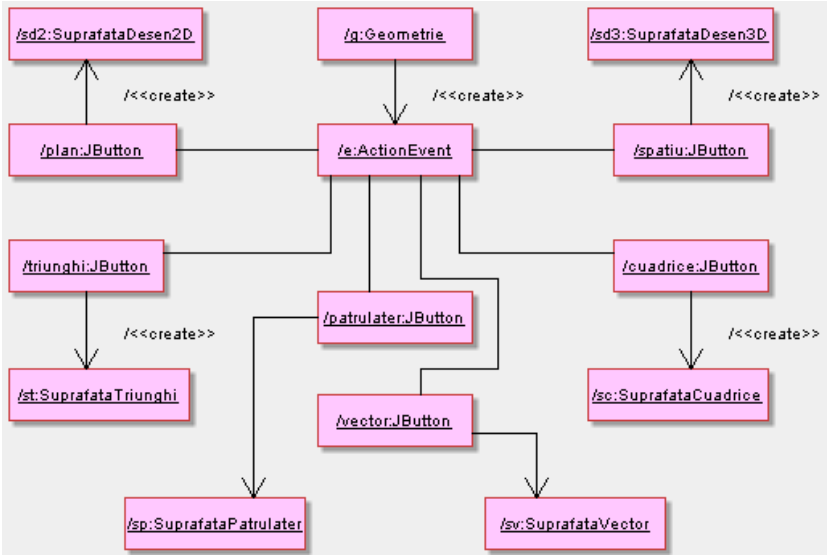


Figura 3.4.28. Diagramă de colaborare pentru obiectele care compun interfața principală

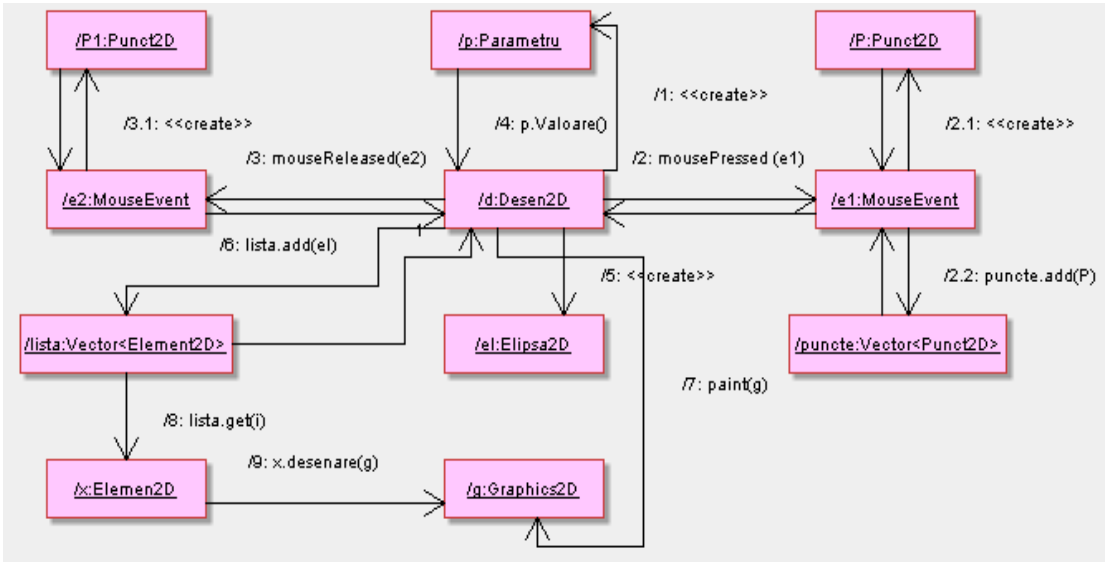


Figura 3.4.29. Diagramă de colaborare pentru desenarea unei elipse

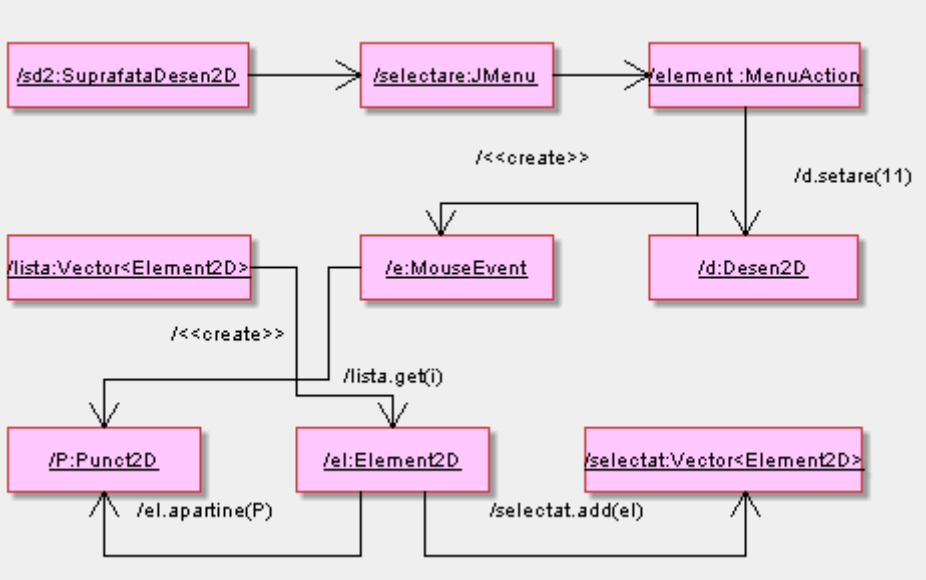


Figura 3.4.30. Diagramă de colaborare pentru selectarea unui element

Diagrama prezentată în figura 3.4.29 redă interacțiunile dintre obiecte care au ca scop desenarea unei elipse. Se observă că există interacțiuni între obiecte de tip: *Desen2D*, *Parametru, MouseEvent*, *Punct2D*, *Vector<Element2D>, Element2D* şi *Graphics2D*.

Diagrama prezentată în figura 3.4.30 redă interacțiunile dintre obiecte care au ca scop selectarea unui element. Se observă că există interacțiuni între obiecte de tip: *SuprafataDesen2D, JMenu, MenuAction, Desen2D, MouseEvent, Element2D, Punct2D şi Vector<Element2D>*.

Diagrama prezentată în figura 3.4.31 redă interacțiunile dintre obiecte care au ca scop desenarea tangentei într-un punct la hiperbolă. Se observă că există interacțiuni între obiecte de tip: *SuprafataDesen2D, JMenu, MenuAction, Desen2D, MouseEvent, Punct2D, JOptionPane, Hiperbola2D, Dreapta2D şi Vector<Element2D>*. Fluxul de control reprezentat în diagramă specifică ordonarea în timp a mesajelor, prin prefixarea fiecărui mesaj cu un număr.

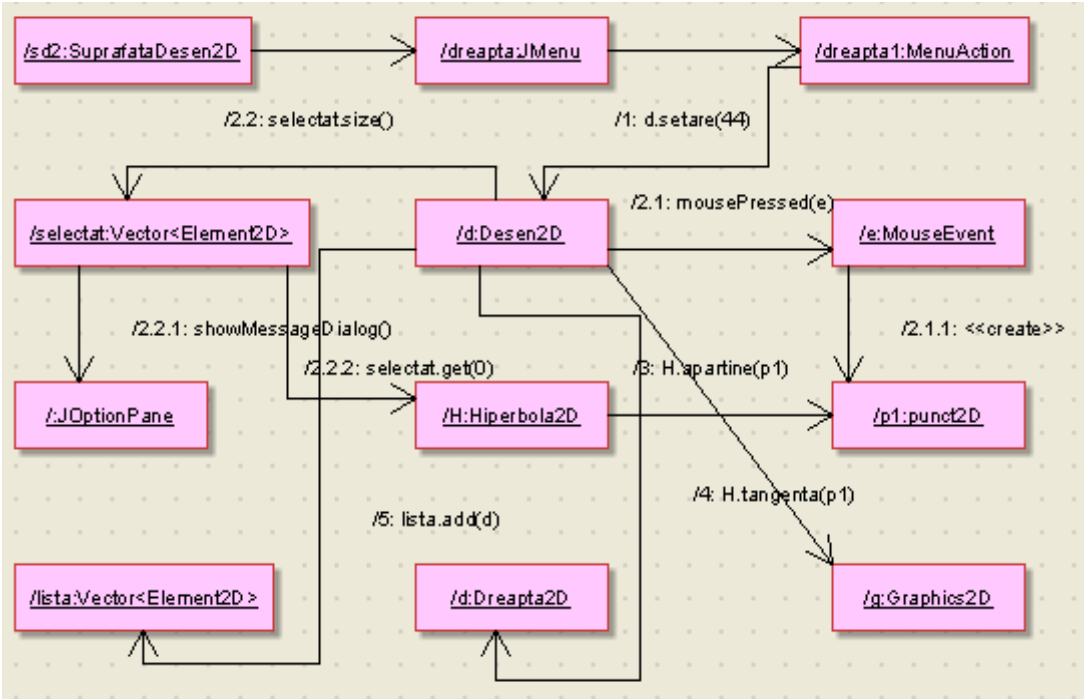


Figura 3.4.31. Diagramă de colaborare pentru desenarea tangentei într-un punct la hiperbolă

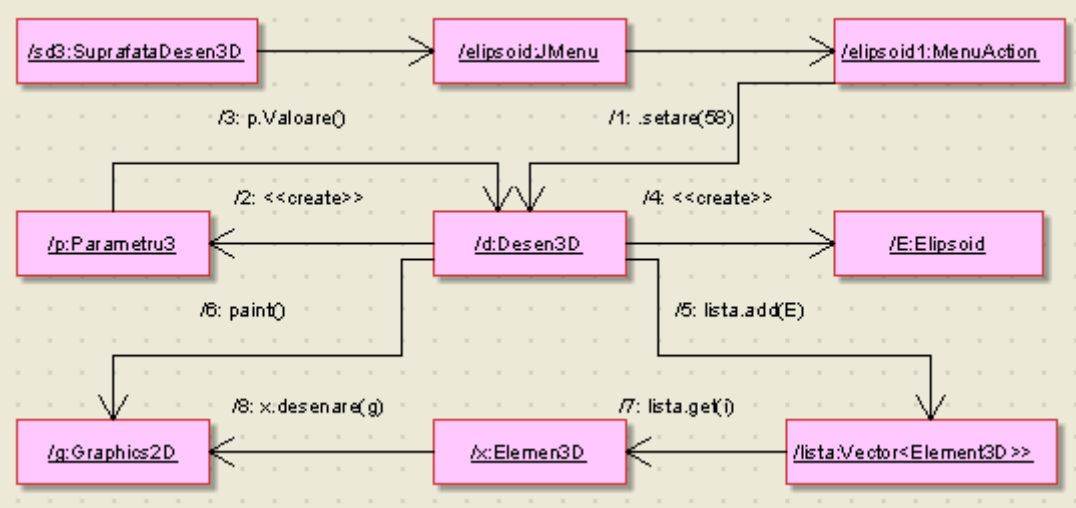


Figura 3.4.32. Diagramă de colaborare pentru desenarea unui elipsoid

Diagrama prezentată în figura 3.4.32 redă interacțiunile dintre obiecte care au ca scop desenarea unui elipsoid. Se observă că există interacțiuni între obiecte de tip: *SuprafataDesen3D, JMenu, MenuAction, Desen3D, Parametru3, Elipsoid, Graphics2D, Element3D şi Vector<Element3D>*. Fluxul de control reprezentat în diagramă specifică ordonarea în timp a mesajelor, prin prefixarea fiecărui mesaj cu un număr.

Diagrama prezentată în figura 3.4.33 redă interacțiunile dintre obiecte care au ca scop desenarea planului tangent la hiperboloid într-un punct. Se observă că există interacțiuni între obiecte de tip: *SuprafataDesen3D, JMenu, MenuAction, Desen3D, Parametru2, Hiperboloid, Graphics2D, Punct3D, JOptionPane şi Vector<Element3D>*. Fluxul de control reprezentat în diagramă specifică ordonarea în timp a mesajelor, prin prefixarea fiecărui mesaj cu un număr.

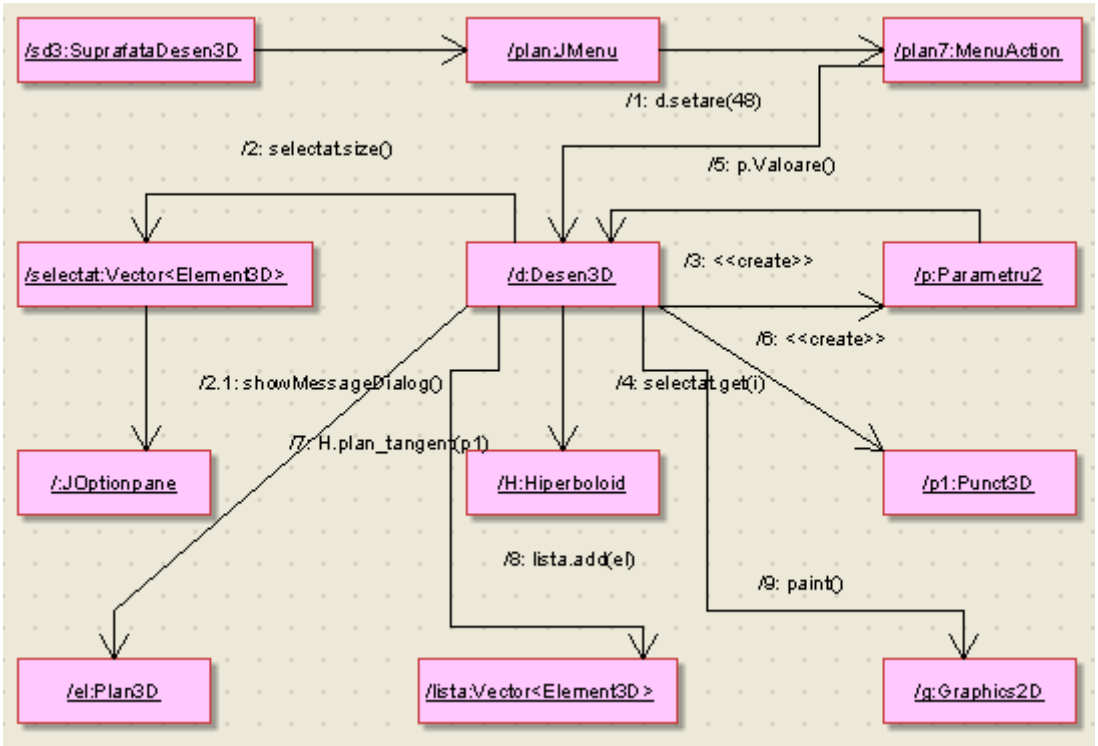


Figura 3.4.33. Diagramă de colaborare pentru desenarea planului tangent la hiperboloid într-un punct

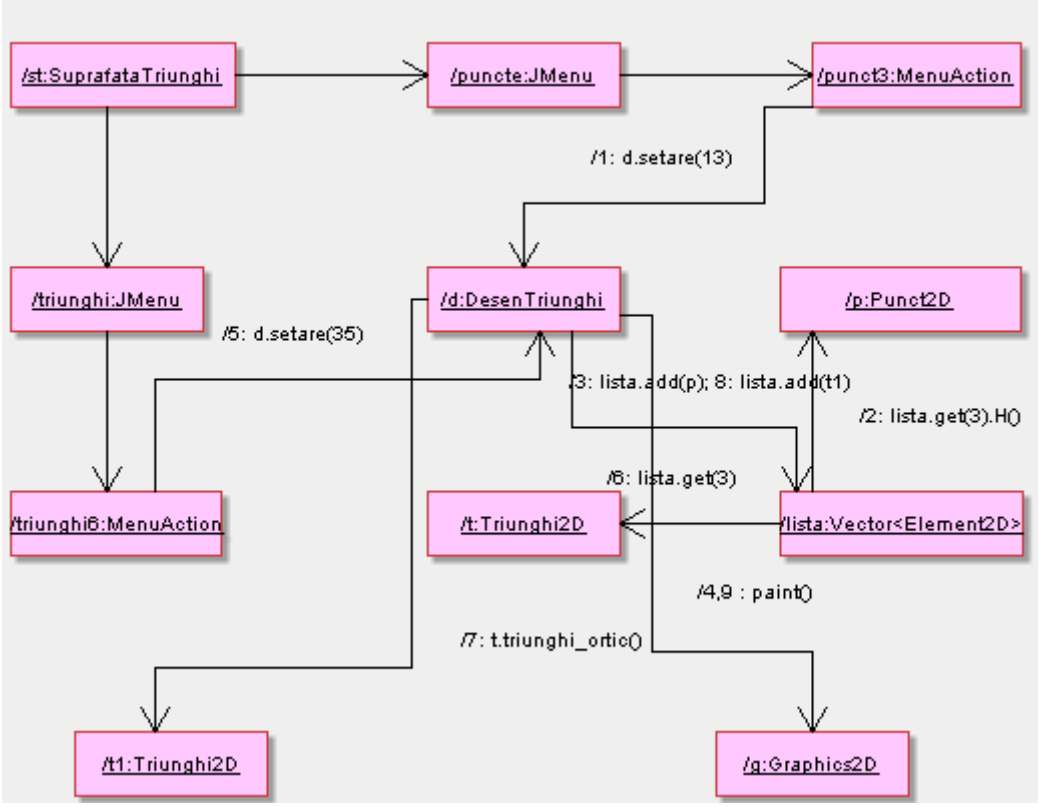


Figura 3.4.34. Diagramă de colaborare pentru desenarea ortocentrului şi triunghiului ortic al unui triunghi

Diagrama prezentată în figura 3.4.34 redă interacțiunile dintre obiecte care au ca scop desenarea ortocentrului şi triunghiului ortic al unui triunghi. Se observă că există interacțiuni între obiecte de tip: *SuprafataTriunghi, JMenu, MenuAction, DesenTriunghi, Punct2D, Triunghi2D, Graphics2D şi Vector<Element2D>*. Fluxul de control reprezentat în diagramă specifică ordonarea în timp a mesajelor, prin prefixarea fiecărui mesaj cu un număr.

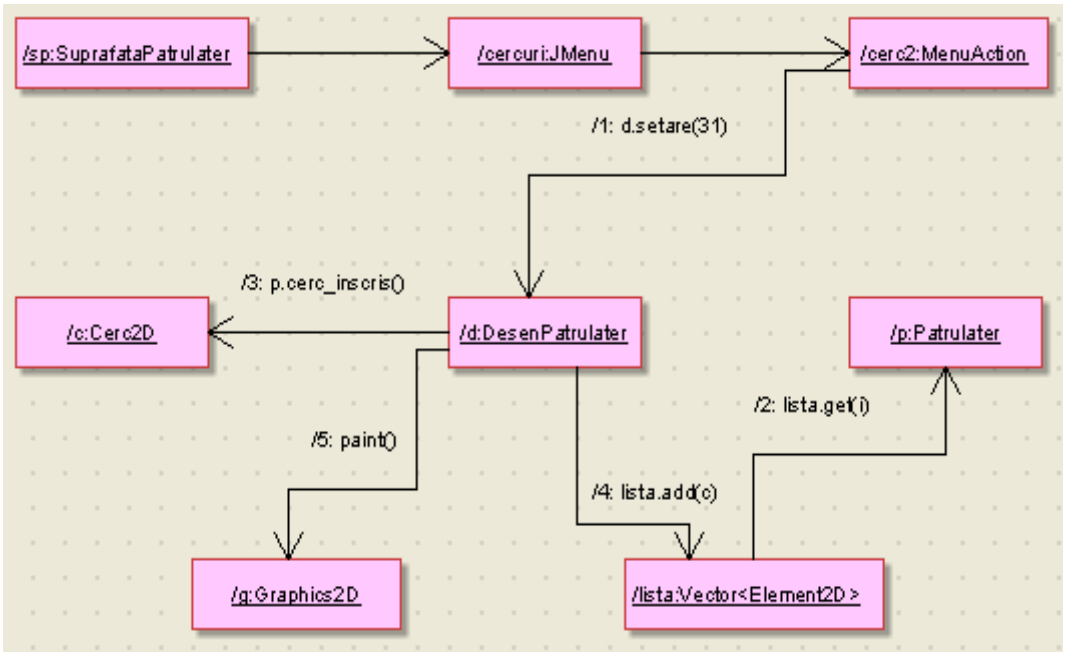


Figura 3.4.35. Diagramă de colaborare pentru desenarea cercului înscris într-un patrulater circumscriptibil

Diagrama prezentată în figura 3.4.35 redă interacţiunile dintre obiecte care au ca scop desenarea cercului înscris într-un patrulater circumscriptibil. Se observă că există interacţiuni între obiecte de tip: *SuprafataPatrulater, JMenu, MenuAction, DesenPatrulater, Patrulater2D, Cerc2D, Graphics2D şi Vector<Element2D>*. Fluxul de control reprezentat în diagramă specifică ordonarea în timp a mesajelor, prin prefixarea fiecărui mesaj cu un număr.

3.4.8 Compararea diagramelor de interacţiune

Pentru sistemul prezentat, modelarea fluxurilor de control s-a realizat, prin intermediul diagramelor de interacţiune [34], în două moduri:

- prin ordonarea în timp, utilizându-se diagramele de secvenţă;
- prin organizare, utilizându-se diagramele de colaborare.

Pentru că au modelat acelaşi tip de comportament, diagramele de secvenţă şi diagramele de colaborare sunt echivalente. Ca urmare, conversiile de la un tip de diagramă la altul sunt posibile, fără a pierde informaţie esenţială.

Acest aspect poate fi probat prin compararea diagramelor prezentate în figurile 3.4.25 şi 3.4.32, diagrame ce prezintă fluxul de control pentru desenarea unui elipsoid.

Diagramele de secvenţă se deosebesc de diagramele de colaborare prin următoarele caracteristici:

- Obiectele ce apar în diagramă au durata de viaţă atât timp cât există interacţiunea, aceasta fiind specificată prin linia vieţii obiectului.
- Are loc focalizarea controlului pe un obiect.

Diagramele de colaborare au două caracteristici care le deosebesc de diagramele de secvenţă:

- Indică modul în care un obiect este legat de altul.
- Poate avea ataşat un sistem de secvenţare prin prefixarea mesajelor cu numere unice. În cazul în care avem mesaje imbricate putem utiliza sistemul prezentat în figura 3.4.29: după mesajul numărul 2 avem imbricate două mesaje notate 2.1 şi 2.2.

3.5 Faza de implementare

Diagrama componentelor este asemănătoare cu diagrama pachetelor, permiţând vizualizarea modului în care sistemul este divizat şi a dependenţelor dintre module [42,79]. Diagrama componentelor pune însă accentul pe elementele *software* fizice (fişiere, biblioteci, executabile) şi nu pe elementele logice, ca în cazul pachetelor.

Diagrama din figura 3.5.1 descrie colecţia de componente care, împreună, asigură funcţionalitatea pentru partea sistemului informatic interactiv ce permite realizarea construcţiilor geometrice în plan. Componenta centrală a diagramei *SuprafataDesen2D.class*, componentă obţinută prin transformarea de către compilatorul Java a componentei *SuprafataDesen2D.java* în cod executabil. După cum se observă această componentă interacţionează direct doar cu componentele *Desen2D.class* şi *MenuAction.class*.

Componenta *Desen2D.class* interacţionează direct doar cu componentele *Element2D.class* şi *Parametru.class*, pe când componenta *Element2D.class* interacţionează cu toate fişierele ce implementează clasele corespunzătoare elementelor geometrice din plan şi transformările acestora. Componenta *Izometrie2D.class* interacţionează cu fişierele ce implementează izometriile în plan: *Simetrie2D.class*, *Translatie2D.class* şi *Rotatie2D.class*.

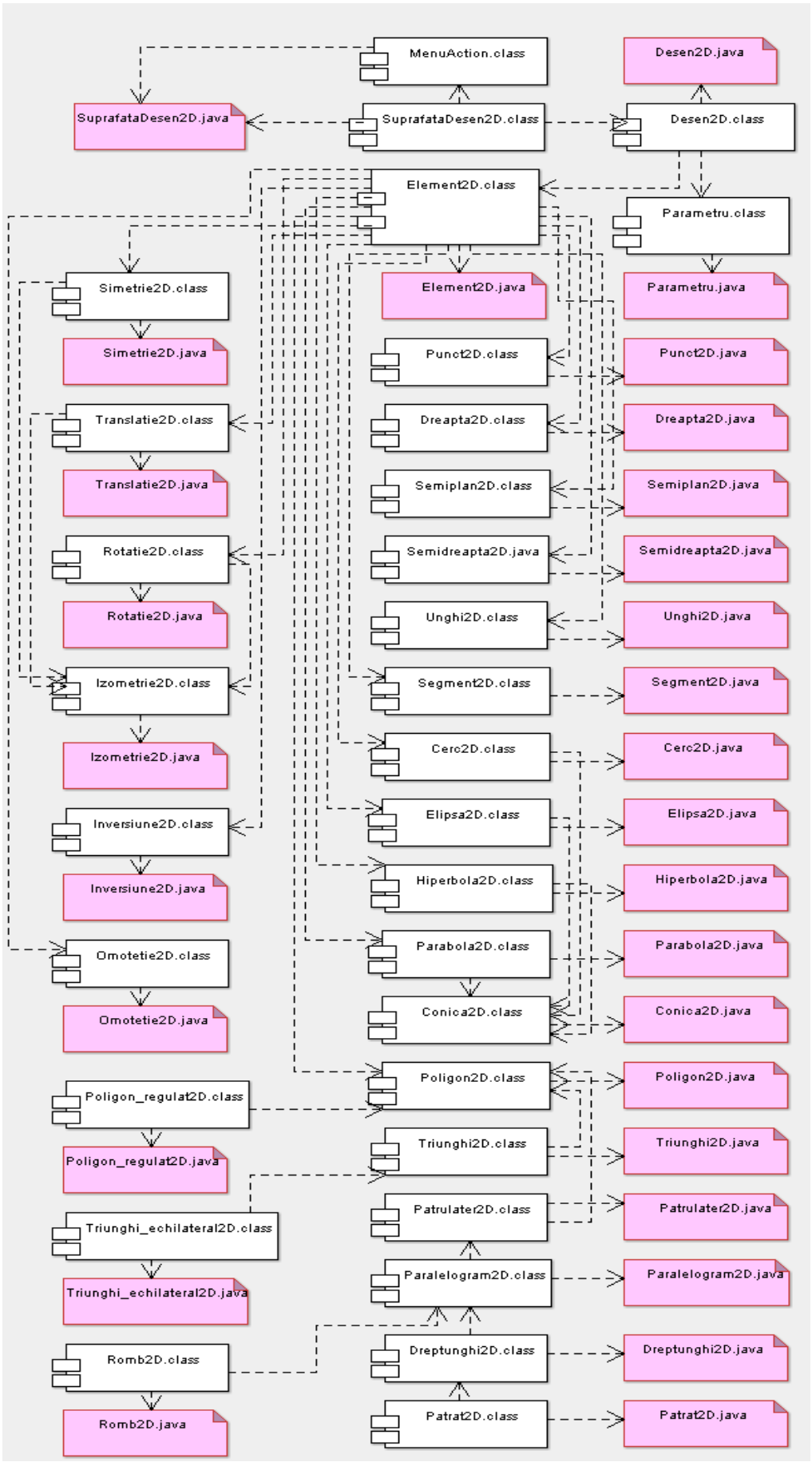


Figura 3.5.1. Diagramă de componente pentru realizarea construcțiilor geometrice în plan

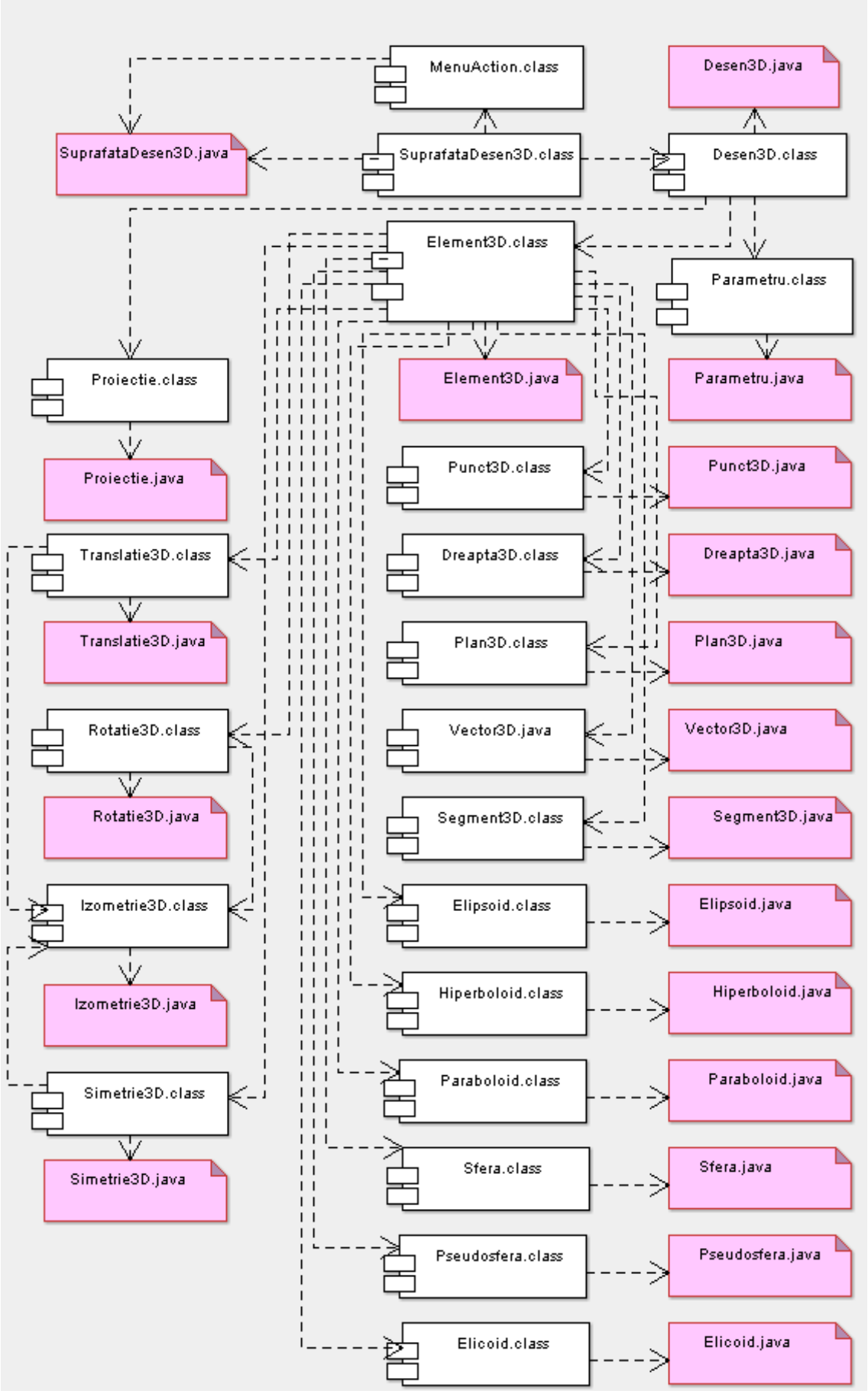


Figura 3.5.2. Diagramă de componente pentru realizarea construcțiilor geometrice în spațiu

Diagrama din figura 3.5.2 descrie colecția de componente care, împreună, asigură funcționalitatea pentru partea sistemului informatic interactiv ce permite realizarea construcțiilor geometrice în spațiu. Componenta centrală a diagramei *SuprafataDesen3D.class*, componentă obținută prin transformarea de către compilatorul Java a componentei *SuprafataDesen3D.java* în cod executabil. După cum se observă această componentă interacționează direct doar cu componentele *Desen3D.class* şi *MenuAction.class*.

Componenta *Desen3D.class* interacționează direct doar cu componentele *Element3D.class, Proiectie.class* şi *Parametru.class*, pe când componenta *Element3D.class* interacționează cu toate fişierele ce implementează clasele corespunzătoare elementelor geometrice din spațiu şi transformările acestora. Componenta *Izometrie3D.class* interacționează cu fişierele ce implementează izometriile în spațiu: *Simetrie3D.class*, *Translatie3D.class* şi *Rotatie3D.class*.

3.6 Concluzii

Prin reprezentarea diagramelor corespunzătoare celor trei faze: analiză, proiectare şi implementare sistemul informatic interactiv a fost descris într-o manieră clară şi concisă. Utilizarea limbajului de modelare UML în realizarea diagramelor este caracterizată de rigoare sintactică, semantică bogată şi suport pentru modelarea vizuală.

Diagramele au fost realizate printr-o abordare într-o manieră nouă, multidisciplinară a aplicației informatice, înglobând atât metodele pedagogiei moderne, cât şi componentele specifice disciplinei de studiat. S-a realizat astfel legătura dintre acțiunile didactice şi scopurile şi obiectivele ştiințific stabilite, prin elaborarea a noi metode şi prin asimilarea a noi mijloace, capabile să sporească randamentul şcolar, permițând elevilor şi studenților să-şi însuşească sistemul cerut de cunoştințe şi tehnici de aplicare a acestora în condiții cât mai optime.

Comparând diagramele prezentate anterior cu descrierea atelierului virtual prezentat în [70], se constată că modelarea sistemul interactiv permite:

- Definirea concretă a fiecărui caz de utilizare prin prezentarea diagramelor de activități;
- Implementarea eficientă a claselor corespuzătoare tuturor elementelor geometrice prin realizarea de relații de moştenire, agregare şi compunere;
- Descrierea stărilor prin care trece un obiect şi a tranzițiilor dintre aceste stări;
- Descrierea interacțiunilor dintre obiecte în contexte diferite prin realizarea diagramelor de colaborare, diagrame ce nu sunt prezentate în cazul atelierului virtual anterior amintit;
- Vizualizarea modului în care sistemul este divizat şi a dependențelor dintre module prin realizarea diagramelor de componente, diagrame inexistente în cazul atelierului virtual pentru geometrie amintit anterior.

ARHITECTURA SISTEMULUI INFORMATIC EDUCAȚIONAL

4.1 JAVA – limbaj de programare ales

Pentru orice proiect *software* limbajul de programare utilizat influențează întregul proiect. Această alegere este o decizie mare în proiectare şi trebuie luată cu mare grijă. În acest paragraf vor fi prezentate motivele pentru care am ales Java şi de ce această alegere poate fi corectă şi pentru alte proiecte.

S-a urmărit alegerea unui limbaj independent de arhitectura calculatorului pe care lucrează şi foarte portabil. În loc să genereze cod nativ pentru o platformă sau alta, compilatorul Java generează o secvență de instrucțiuni ale unei maşini virtuale Java [20]. Execuția aplicațiilor Java este interpretată. Singura parte din mediul de execuție Java care trebuie portată de pe o arhitectură pe alta este mediul de execuție cuprinzând interpretorul şi o parte din bibliotecile standard care depind de sistem. În acest fel, aplicații Java compilate pe o arhitectură SPARC de exemplu, pot fi rulate fără recompilare pe un sistem bazat pe procesoare Intel.

Java este un limbaj simplu de folosit chiar şi de către programatorii neprofesionişti, programatori care doresc să se concentreze asupra aplicațiilor în principal şi abia apoi asupra tehnicilor de implementare a acestora. Această trăsătură poate fi considerată ca o reacție directă la complexitatea considerabilă a limbajului C++. Aceasta este una din cele mai importante caracteristici ale limbajului foarte des subestimată. Este uşor să se scrie proiecte complexe în Java pornind doar de la schițe.

Limbajul Java [71] este în totalitate orientat obiect. Cu el se pot crea clase de obiecte şi instanțe ale acestora, se pot încapsula informațiile, se pot moşteni variabilele şi metodele de la o clasă la alta etc. Singura trăsătură specifică limbajelor orientate obiect care lipseşte este moştenirea multiplă, dar pentru a suplini această lipsă, Java oferă o facilitate mai simplă, numită interfață, care permite definirea unui anumit comportament pentru o clasă de obiecte, altul decât cel definit de clasa de bază. Era necesară alegerea unui limbaj orientat obiect pentru a putea crea clase corespunzătoare obiectelor geometrice în plan şi în spațiu.

4.2 Structuri de date optimizate pentru geometria dinamică

Proiectarea structurilor de date este crucială pentru stabilitatea şi performanța întregului sistem interactiv. Dacă ne gândim la ierarhii de clase pentru un sistem orientat obiect pentru geometrie dinamică, vom sfârşi prin a reprezenta moştenirea ca în arborii din figurile 3.3.1 – 3.3.6 prezentate în capitolul anterior. Clasa de bază pentru toate obiectele geometrice în plan este *Element2D*, iar pentru obiectele geometrice în spațiu este *Element3D*. Aceste două clase conțin toate metodele şi atributele ce sunt utilizate pentru toate elementele geometrice, cum ar fi, spre exemplu, culoarea şi stilul de desenare al unui element geometric, şi metode de desenare ale acestuia.

Subclasele clasei *Element2D*, cum ar fi: *Punct2D*, *Dreapta2D*, *Segment2D*, *Vector2D*, *Poligon2D*, *Conica2D*, conțin date adiționale ce sunt necesare pentru respectivul element

geometric (Anexele 1 şi 2), cum ar fi: coordonatele unui punct, panta unei drepte, dreapta suport pentru un segment, vârfurile şi laturile unui poligon, focarele unei elipse sau hiperbole, dreapta directoare a unei parabole. Subclasele clasei *Element3D*, cum ar fi: *Punct3D*, *Dreapta3D*, *Segment3D*, *Vector3D*, *Plan3D*, *Elipsoid, Hiperboloid, Paraboloid, Sfera, Pseudosfera, Elicoid*, conţin date adiţionale ce sunt necesare pentru respectivul element geometric (Anexele 1 şi 2), cum ar fi: coordonatele unui punct, extremităţile unui segment, originea şi extremitatea unui vector, coeficienţii din ecuaţia unui plan în spaţiu.

Pe lângă aceste clase corespunzătoare noţiunilor geometrice, există clase ce conţin algoritmi pentru reprezentarea atât a elementelor geometrice independente, cât şi a celor dependente, cum ar fi: intersecţia a două drepte neparalele, cercul circumscris unui triunghi, tangenta şi normala la o conică într-un punct, intersecţia a două plane neparalele, vectorul normal al unui plan, planul tangent şi normala la o cuadrică.

Clasa *Desen2D* conţine algoritmi pentru reprezentarea construcţiilor geometrice în plan, iar clasa *Desen3D* pentru desenare în spaţiu. În cazul prezentării noţiunilor teoretice şi a unor aplicaţii ale acestora se folosesc următoarele clase: clasa *DesenTriunghi* pentru capitolul *Triunghiuri*, clasa *DesenPatrulater* pentru capitolul *Patrulatere*, clasa *DesenVector* pentru capitolul *Vectori în plan* şi clasa *DesenCuadrice* pentru capitolul *Cuadrice*.

În figura 4.2.1 este prezentată o parte din structura clasei *Desen2D* ce cuprinde o listă cu toate elementele geometrice create, care sunt instanţe ale clasei *Element2D*. Algoritmii operează cu datele ce sunt stocate prin intermediul atributelor instanţelor.

Primul algoritm prezentat primeşte ca date de intrare două elemente de tip *Punct2D* şi returnează o instanţă a clasei *Dreapta2D* determinată prin datele de intrare. Algoritmul de intersecţie primeşte tot două date de intrare: o instanţă a clasei *Dreapta2D*, anterior creată, şi o instanţă a clasei *Cerc2D*. Dacă dreapta este secantă cercului, atunci algoritmul va returna, ca date de ieşire, două instanţe ale clasei *Punct2D* ce reprezintă punctele de intersecţie.

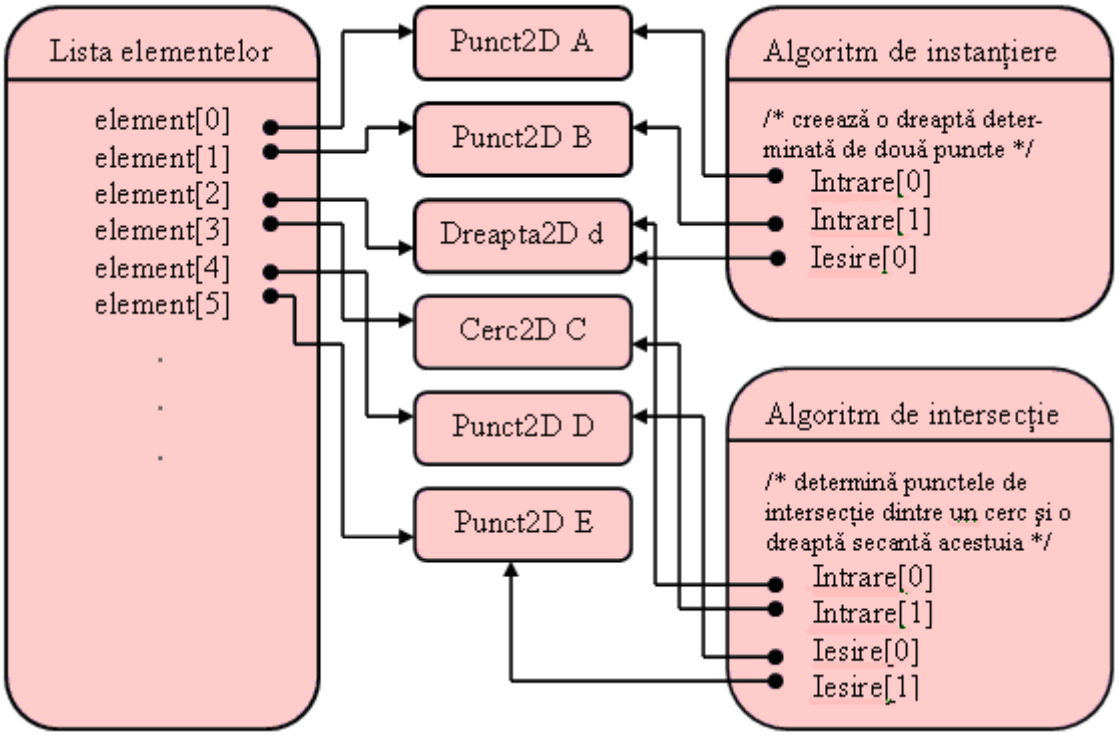


Figura 4.2.1. Parte din structura unei instanţe a clasei *Desen2D*

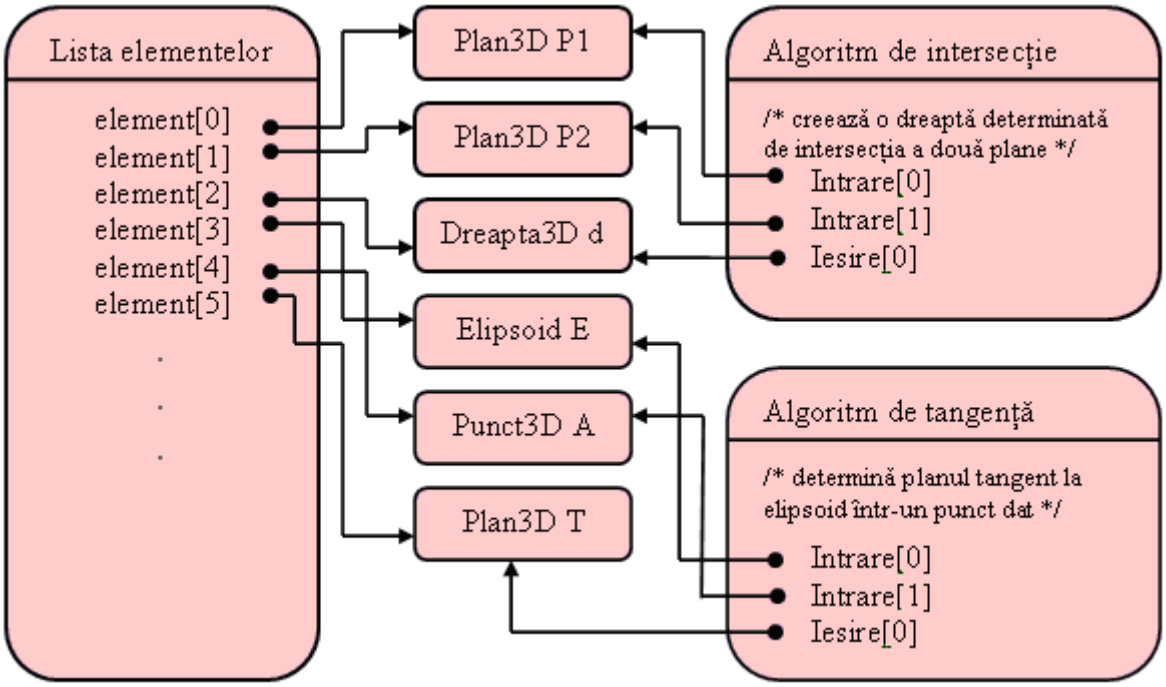


Figura 4.2.2. Parte din structura unei instanţe a clasei *Desen3D*

În figura 4.2.2 este prezentată o parte din structura clasei *Desen3D* ce cuprinde o listă cu toate elementele geometrice create, care sunt instanţe ale clasei *Element3D*. Algoritmii operează cu datele ce sunt stocate prin intermediul atributelor instanţelor.

Primul algoritm prezentat primeşte ca date de intrare două elemente de tip *Plan3D* şi returnează o instanţă a clasei *Dreapta3D* ce reprezintă intersecţia celor două plane neparalele. Algoritmul de tangenţă primeşte tot două date de intrare: o instanţă a clasei *Elipsoid* şi o instanţă a clasei *Punct3D*. Algoritmul va returna, ca dată de ieşire, o instanţă ale clasei *Plan3D* ce reprezintă planul tangent la elipsoid.

Algorimii utilizaţi pentru implementare utilizează metode speciale de programare [63], contribuind la rapiditatea şi exactitatea reprezentării grafice a elementelor geometrice dorite.

4.3 Interfaţa grafică a sistemului informatic

Interfaţa grafică permite manipularea sistemului într-o manieră foarte avantajoasă şi simplă. În continuare vor fi prezentate caracteristicile ferestrelor corespunzătoare celor două cazuri de realizare de construcţii geometrice, în plan şi în spaţiu, precum şi a celor patru cazuri de prezentare de noţiuni şi rezultate.

În figura 4.3.1 este prezentată bara de meniuri a interfeţei ce permite realizarea de construcţii geometrice exacte în plan. Aceasta cuprinde patru meniuri: *Fisier*, *Editare*, *Vizualizare* şi *Geometrie plană*.

Meniul *Fisier* cuprinde patru opţiuni: crearea unei noi suprafeţe de desenare, salvarea unei construcţii geometrice, tipărirea construcţiei geometrice şi închiderea interfeţei grafice.

Meniul *Editare* cuprinde două opţiuni şi două submeniuri. Prima opţiune permite ştergerea ultimului element geometric desenat prin revenire la pasul anterior, iar a doua opţiune permite ştergerea tuturor elementelor selectate în acel moment. Submeniul *Selectare* cuprinde două opţiuni pentru selectarea unui element din construcţie şi selectarea tuturor elementelor create. Al

doilea submeniu *Stil_desenare* conține la rândul său alte trei submeniuri ce permit selectarea stilului de desenare (patru opțiuni), a grosimii liniei (cinci posibilități) şi a culorii dorite. În figură sunt prezentate cele zece culori ce pot fi selectate.

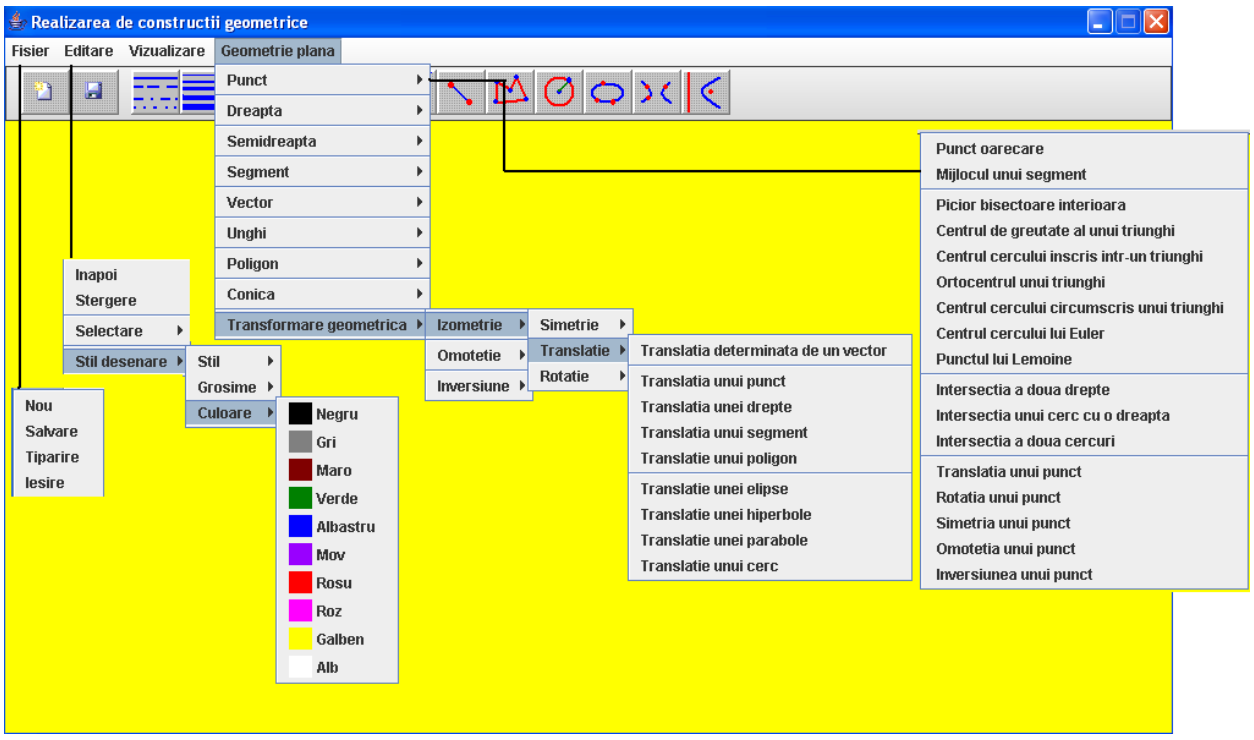


Figura 4.3.1. Meniul interfeței grafice corespunzătoare opțiunii de realizare a construcțiilor geometrice în plan

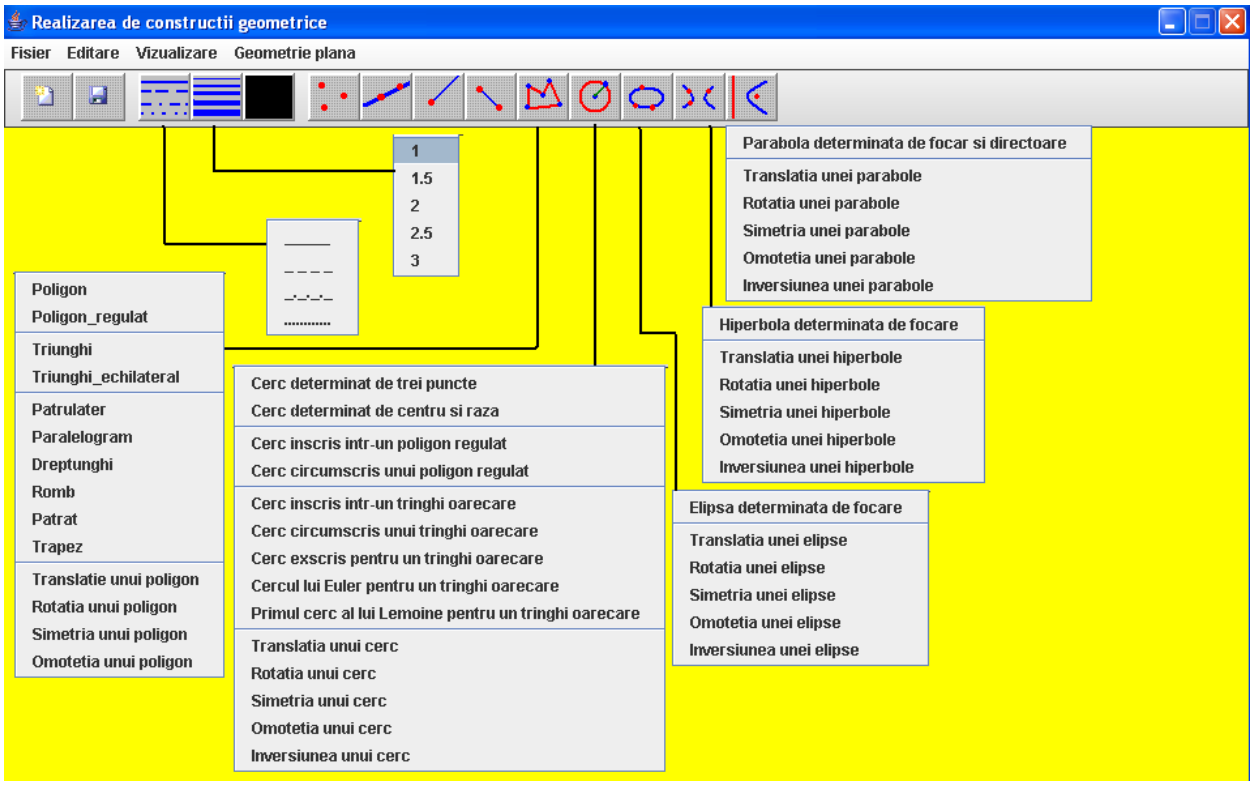


Figura 4.3.2. Bara de instrumente din interfața grafică corespunzătoare opțiunii de realizare a construcțiilor geometrice în plan

Al treilea meniu *Vizualizare* cuprinde variante de afişare a construcţiei geometrice cum ar fi reprezentarea sistemului ortogonal de axe.

Ultimul meniu, şi cel mai complex, cuprinde la rândul său opt submeniuri ce vor permite crearea elementelor geometrice şi un meniu pentru transformările geometrice din plan. Dintre primele opt submeniuri este prezentat în figura 4.3.1 doar submeniul ce permite crearea de elemente geometrice de tip *Punct2D* în funcţie de opţiunea aleasă. Se poate observa că se poate determina mijlocul unui segment, centrul de greutate şi ortocentrul unui triunghi, punctul de intersecţie a două drepte sau punctul obţinut prin aplicarea unei transformări geometrice.

Submeniul corespunzător transformărilor geometrice conţine trei submeniuri pentru crearea de izometrii, omotetii şi inversiuni. Submeniul *Izometrie* cuprinde trei submeniuri pentru crearea de rotaţii, simetrii şi translaţii. Cele şapte meniuri corespunzătoare transformărilor geometrice conţin, pe lângă opţiuni de instanţiere de obiecte de acest tip, opţiuni ce returnează elemente geometrice obţinute în cazul aplicării transformărilor geometrice obiectelor de tip Punct2D, Dreapta2D, Segment2D, Poligon2D, Elipsa2D, Hiperbola2D, Parabola2D şi Cerc2D.

În figura 4.3.2 este prezentată bara de instrumente a interfeţei ce permite realizarea de construcţii geometrice exacte în plan. Aceasta cuprinde trei grupuri de butoane corespunzătoare operaţiilor cele mai utilizate. Primul grup cuprinde un buton pentru crearea unei noi suprafeţe de desenare şi un buton pentru salvarea unei construcţii geometrice. Al doilea grup este format din trei butoane şi conţine opţiunile din submeniul *Stil_desenare* al meniului *Editare*. În figură sunt prezentate opţiunile pentru primele două butoane. Ultimul grup este utilizat pentru crearea de elemente geometrice şi corespunnde unor opţiunii din primele opt submeniuri ale meniului *Geometrie plana*.

În figura 4.3.3 este prezentată bara de meniuri a interfeţei ce permite realizarea de construcţii geometrice exacte în spaţiu. Aceasta cuprinde patru meniuri: *Fisier*, *Editare*, *Vizualizare* şi *Geometrie analitica în spatiu*.

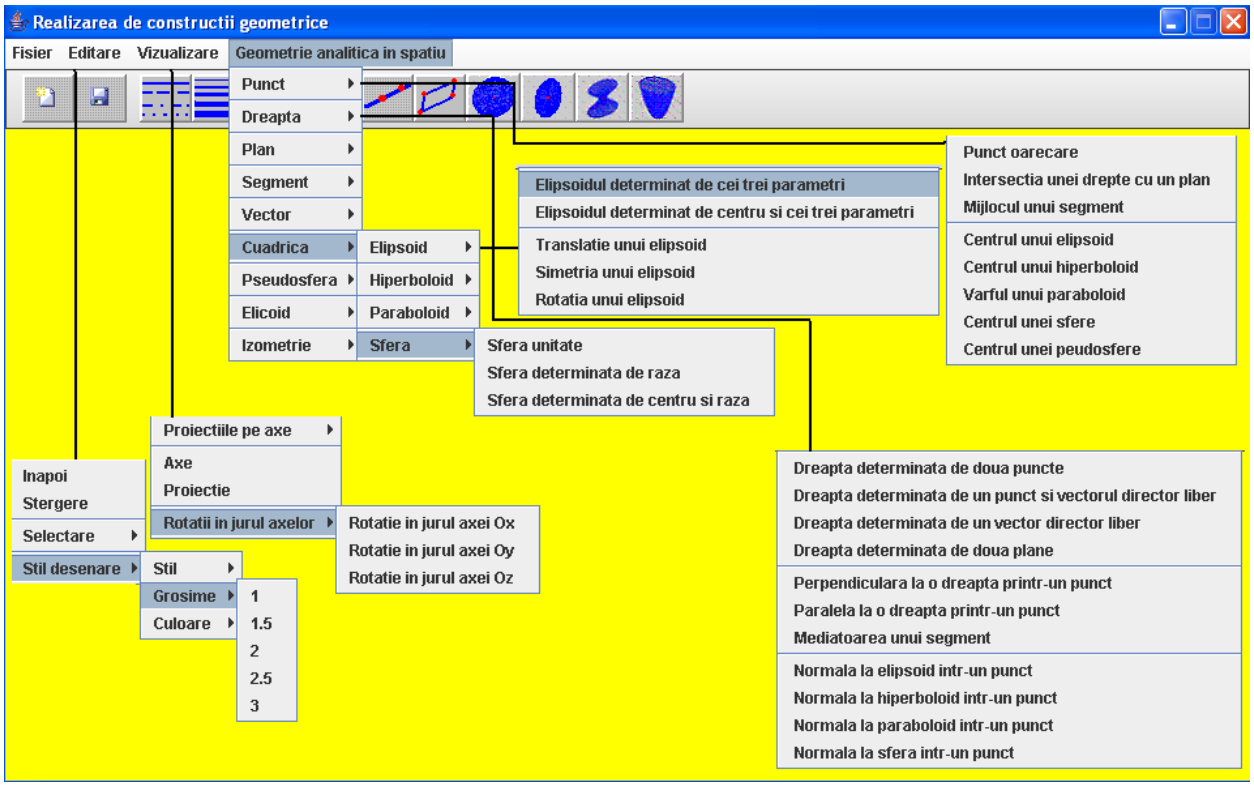


Figura 4.3.3. Meniul interfeţei grafice corespunzătoare opţiunii de realizare a construcţiilor geometrice în spaţiu

Meniurile *Fisier* şi *Editare* coincid cu cele din cazul anterior. Al treilea meniu *Vizualizare* cuprinde variante de afişare a construcţiei geometrice cum ar fi reprezentarea sistemului ortogonal de axe, tipul de proiecţie utilizat sau vizualizarea construcţiei geometrice din mai multe puncte de observare.

Ultimul meniu, şi cel mai complex, cuprinde la rândul său opt submeniuri ce vor permite crearea elementelor geometrice şi un meniu pentru izometrii în spaţiu. Dintre primele opt submeniuri sunt prezentate în figura 4.3.3 doar submeniurile ce permit crearea de elemente geometrice de tip *Punct3D* şi *Dreapta3D* în funcţie de opţiunea aleasă. Se poate observa că se poate determina mijlocul şi mediatoarea unui segment, punctul de intersecţie dintre o dreaptă şi un plan, centrul unui elipsoid, paralela şi perpendiculara la o dreaptă printr-un punct, normala la o cuadrică.

Submeniul *Cuadrica* este format din patru submeniuri corespunzătoare elementelor geometrice de tip elipsoid, hiperboloid, paraboloid şi sfera. În figură sunt prezentate primul şi ultimul submeniu. Submeniul *Izometrie* cunprinde trei submeniuri pentru crearea de rotaţii, simetrii şi translaţii.

În figura 4.3.4 este prezentată bara de meniuri a interfeţei ce permite prezentarea noţiunilor şi rezultatelor principale corespunzătoare capitolului *Triunghi*. Aceasta cuprinde şapte meniuri: *Fisier*, *Vizualizare, Puncte, Linii importante, Triunghiuri, Cercuri* şi *Teoreme*, dintre care ultimele cinci sunt prezentate în figura anterioară.

Meniul *Fisier* cuprinde patru opţiuni: crearea unei noi suprafeţe de desenare, salvarea unei construcţii geometrice, tipărirea construcţiei geometrice şi închiderea interfeţei grafice. Al doilea meniu *Vizualizare* cuprinde variante de afişare a construcţiei geometrice cum ar fi reprezentarea sistemului ortogonal de axe sau prezentarea teoriei specifice opţiunii alese.

Meniul *Puncte* permite determinarea unor puncte caracteristice pentru un triunghi, cum ar fi: ortocentrul triunghiului, centrul cercului înscris, centrul cercului lui Euler, punctul lui Lemoine sau punctul lui Gergonne.

Prin selectarea unei opţiuni a meniului *Linii importante* se reprezintă grafic segmentul sau dreapta specifică opţiunii alese, cum ar fi: simedianele triunghiului, liniile mijlocii, dreapta ortică, dreapta lui Simson sau dreapta lui Lemoine.

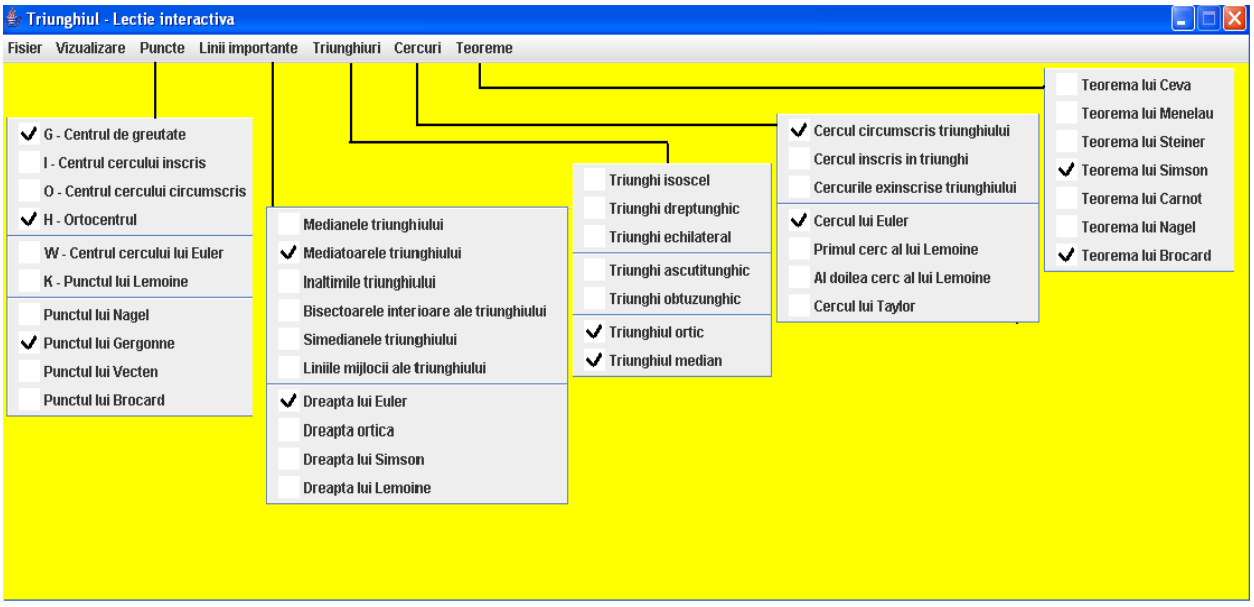


Figura 4.3.4. Meniul interfeţei grafice corespunzătoare opţiunii de prezentare a capitolului *Triunghi*

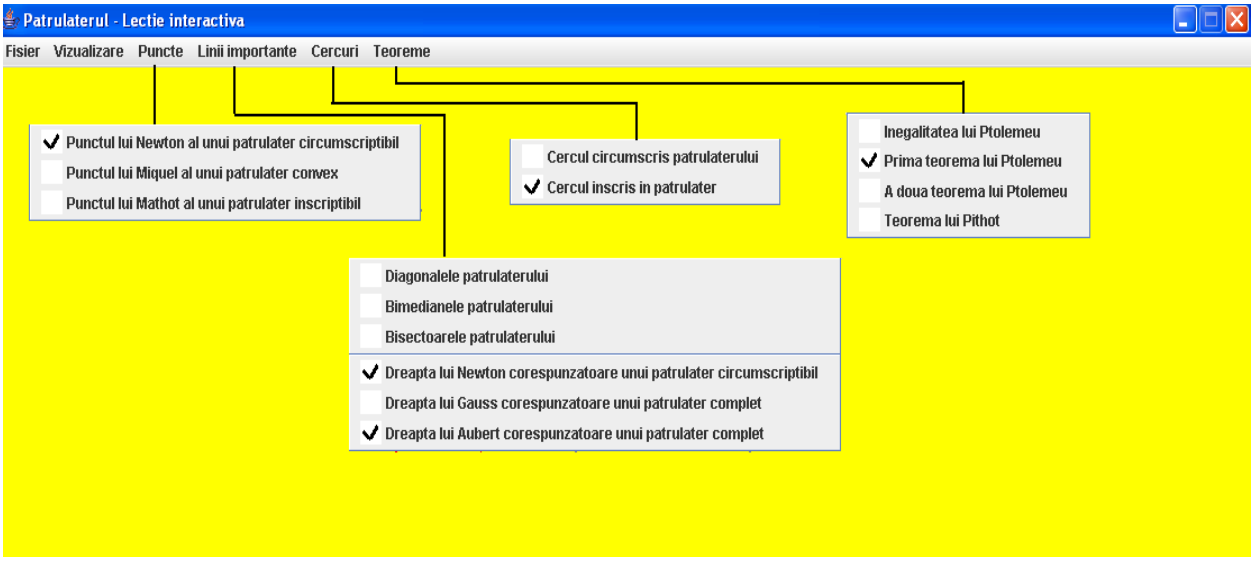


Figura 4.3.5. Meniul interfeței grafice corespunzătoare opțiunii de prezentare a capitolului *Patrulater*

Opțiunile meniului *Triunghiuri* permit verificarea tipului triunghiului existent şi desenarea triunghiului ortic, precum şi a celui median. Meniul *Cercuri* permite determinarea cercurilor caracteristice pentru un triunghi, cum ar fi: cercul circumscris, cercul înscris, cercul lui Euler, cercurile lui Lemoine sau cercul lui Taylor. Ultimul meniu permite prezentarea unor teoreme specifice triunghiurilor, cum ar fi: teorema lui Ceva, teorema lui Menelau, teorema lui Simson, teorema lui Carnot sau teorema lui Brocard.

În figura 4.3.5 este prezentată bara de meniuri a interfeței ce permite prezentarea noțiunilor şi rezultatelor principale corespunzătoare capitolului *Patrulater*. Aceasta cuprinde şase meniuri: *Fisier*, *Vizualizare, Puncte, Linii importante, Cercuri* şi *Teoreme*, dintre care ultimele patru sunt prezentate în figura anterioară. Meniurile *Fisier* şi *Editare* coincid cu cele din cazul anterior.

Meniul *Puncte* permite determinarea unor puncte caracteristice pentru un patrulater, cum ar fi: punctul lui Newton al unui patrulater circumscristibil, punctul lui Miquel al unui patrulater convex sau punctul lui Mathot al unui patrulater inscriptibil.

Prin selectarea unei opțiuni a meniului *Linii importante* se reprezintă grafic segmentul sau dreapta specifică opțiunii alese, cum ar fi: bimedianele patrulaterului, dreapta lui Newton corespunzătoare unui patrulater circumscriptibil sau dreapta lui Aubert corespunzătoare unui patrulater complet.

Meniul *Cercuri* permite determinarea cercului circumscris unui patrulater inscriptibil şi a cercului înscris într-un patrulater circumscriptibil. Ultimul meniu permite prezentarea unor teoreme specifice patrulaterelor, cum ar fi: teoremele lui Ptolemeu sau teorema lui Pithot.

În figura 4.3.6 este prezentată bara de meniuri a interfeței ce permite prezentarea noțiunilor şi rezultatelor principale corespunzătoare capitolului *Vectori*. Aceasta cuprinde cinci meniuri: *Fisier*, *Vizualizare, Vectori, Operatii de bază şi Aplicații*, dintre care ultimele trei sunt prezentate în figura anterioară. Meniurile *Fisier* şi *Editare* coincid cu cele din cazul anterior.

Meniul *Vectori* permite reprezentarea grafică a vectorilor determinați de origine şi extremitate, de origine şi parametrii directori sau a vectorilor de poziție.

Prin selectarea unei opțiuni a meniului *Operații de bază* se reprezintă grafic vectorul obținut în urma adunării, scăderii sau înmulțirii (produs vectorial) a doi vectori şi se determină produsul scalar a doi vectori, precum şi produsul mixt a trei vectori.

Meniul *Aplicații* prezintă câteva din cele mai importante rezultate în a căror demonstrație au fost utilizate proprietăți ale vectorilor.

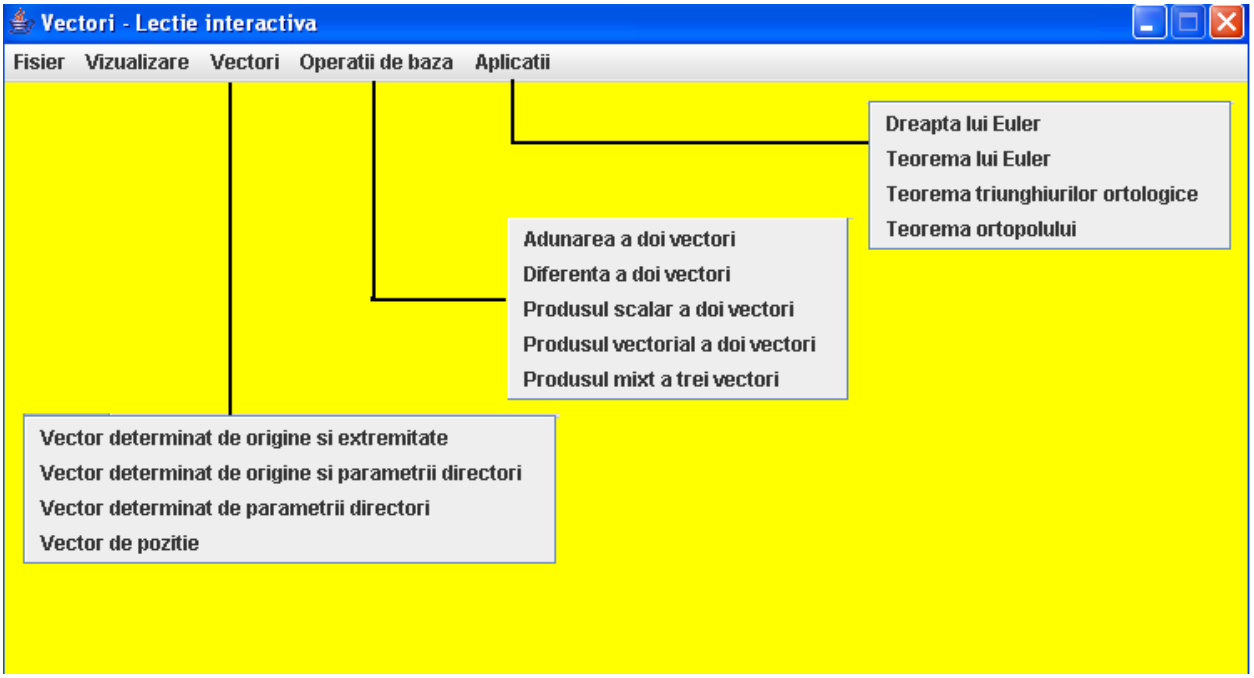


Figura 4.3.6. Meniul interfeței grafice corespunzătoare opțiunii de prezentare a *Vectorilor*

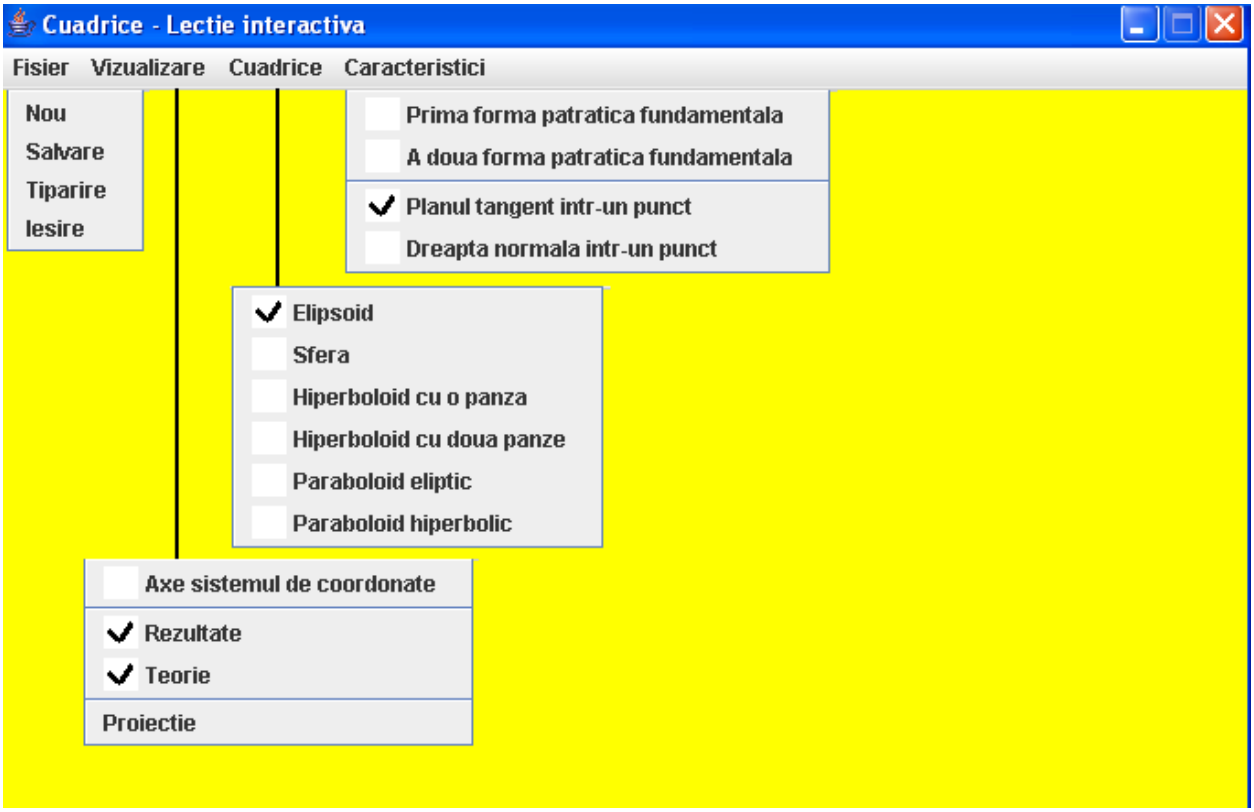


Figura 4.3.7. Meniul interfeței grafice corespunzătoare opțiunii de prezentare a capitolului *Cuadrice*

În figura 4.3.7 este prezentată bara de meniuri a interfeței ce permite prezentarea noțiunilor şi rezultatelor principale corespunzătoare capitolului *Cuadrice*. Aceasta cuprinde patru meniuri: *Fisier*, *Vizualizare, Cuadrice şi Caracteristici*, toate fiind prezentate în figura anterioară.

Meniul *Fisier* coincide cu cel din cazul anterior. Al doilea meniu *Vizualizare* cuprinde variante de afişare a construcției geometrice cum ar fi reprezentarea sistemului ortogonal de axe, tipul de proiecție utilizat sau prezentarea teoriei specifice opțiunii alese.

Meniul *Cuadrice* permite crearea de elemente geometrice ce sunt cazuri particulare de cuadrice. Ultimul meniu cuprinde opțiuni care permit determinarea primei şi celei de-a doua forme pătratice fundamentale specifice tipului de cuadrică reprezentat, precum şi desenarea planului tangent şi a normalei într-un punct al cuadricii.

4.4 Concluzii

În acest capitol s-a realizat o justificare a limbajului de programare ales şi au fost prezentate structurile de date eficiente utilizate pentru geometria dinamică. Tot în acest capitol a fost prezentată interfața grafică a sistemului interactiv, interfața ce va permite manipularea sistemului într-o manieră foarte avantajoasă şi simplă. S-au prezentat caracteristicile ferestrelor corespunzătoare celor două cazuri de realizare de construcții geometrice, în plan şi în spațiu, precum şi a celor patru cazuri de prezentare de noțiuni şi rezultate.

Comparându-se interfața grafică corespunzătoare opțiunii de realizare a construcțiilor geometrice în plan prezentată în figurile 4.3.1 şi 4.3.2 cu interfața grafică a sistemului interactiv Cabri [21], prezentat în capitolul al doilea, se poate observa că noua interfață cuprinde câteva intrumente şi opțiuni ale meniului *Geometrie plana* inexistente în cazul sistemului interactiv Cabri, cum ar fi: triunghi, triunghi_echilateral, patrulater, paralelogram, dreptunghi, romb, pătrat, trapez, cerc determinat de trei puncte, elipsă determinată de focare şi un parametru, hiperbolă determinată de focare şi un parametru, normala şi tangenta într-un punct la o conică, mediana şi simediana unui triunghi corespunzătoare unui vârf, precum şi transformări geometrice aplicate tuturor elementelor geometrice elementare în plan. Față de acelaşi sistem Cabri, se poate observa că sistemul propus în această teză permite realizarea de construcții geometrice în spațiu, interfața grafică fiind prezentată în figura 4.3.3.

Comparându-se aceeaşi interfață grafică corespunzătoare opțiunii de realizare a construcțiilor geometrice în plan prezentată în figurile 4.3.1 şi 4.3.2 cu interfața grafică a sistemului interactiv Cinderella [94], prezentat în capitolul al doilea, se poate observa că noua interfață cuprinde câteva intrumente şi opțiuni ale meniului *Geometrie plana* inexistente în cazul sistemului interactiv Cinderella, cum ar fi: centrul cercului circumscris şi centrul cercului înscris într-un triunghi, ortocentrul şi centrul de greutate al unui triunghi, centrul cercului lui Euler, intersecția a două drepte, mediatoarea unui segment, normala şi tangenta într-un punct la o conică, mediana şi simediana unui triunghi corespunzătoare unui vârf, cercul înscris şi cercul circumscris unui triunghi, cercurile exînscrise pentru un triunghi oarecare, primul cerc al lui Lemoine etc.

Interfața sistemului interactiv permite utilizarea acestuia într-un mod simplu şi elegant atât de către elevi sau studenți, cât şi de cadre didactice. Sistemul informatic prezentat conduce subiectul care îl utilizează la obținerea unei experiențe în înțelegerea şi stăpânirea de cunoştințe din domeniul geometriei şi oferă accesul comod şi eficient la informațiile şi cunoştințele cele mai noi.

FUNCȚIONALITĂȚI ALE SISTEMULUI INFORMATIC EDUCAȚIONAL

În acest capitol vom defini mulțimea operațiilor de bază specifice unui sistem dinamic pentru geometrie. Vor fi descrise explicit cum pot fi efectuate în coordonate carteziene calculele necesare operațiilor de bază.

5.1 Puncte, linii şi poligoane

5.1.1 Reprezentarea dreptelor

Dreptele sunt varietăți liniare de ordinul 1 al planului euclidian [36], definindu-se astfel:

$$\{(x,y)\in\varepsilon^2 \mid a\cdot x+b\cdot y+c=0; a,b\in R; a^2+b^2\neq 0\} \qquad (5.1.1)$$

Două puncte definesc în mod unic o dreaptă. Asftel, utilizând ca date de intrare cele două puncte, poate fi instanțiat un obiect de tip *Dreapta2D* [57, 59]. În figura 5.1.1 sunt reprezentate mai multe drepte determinate de perechi de puncte. Ecuația dreptei determinată de două puncte $A(x_A,y_A)$ şi $B(x_B,y_B)$ este:

$$\frac{x-x_A}{x_B-x_A}=\frac{y-y_A}{y_B-y_A}. \qquad (5.1.2)$$

Altă modalitate de descriere a unei drepte este prin intermediul unui punct şi a pantei [57]. În figura 5.1.2 sunt reprezentate obiecte de tip *Dreapta2D* instanțiate în acest mod. Instanțierea unui obiect de tip *Dreapta2D* se mai poate realiza prin specificarea celor trei parametrii din ecuația generală sau a unui punct aparținând dreptei şi a vectorului director.

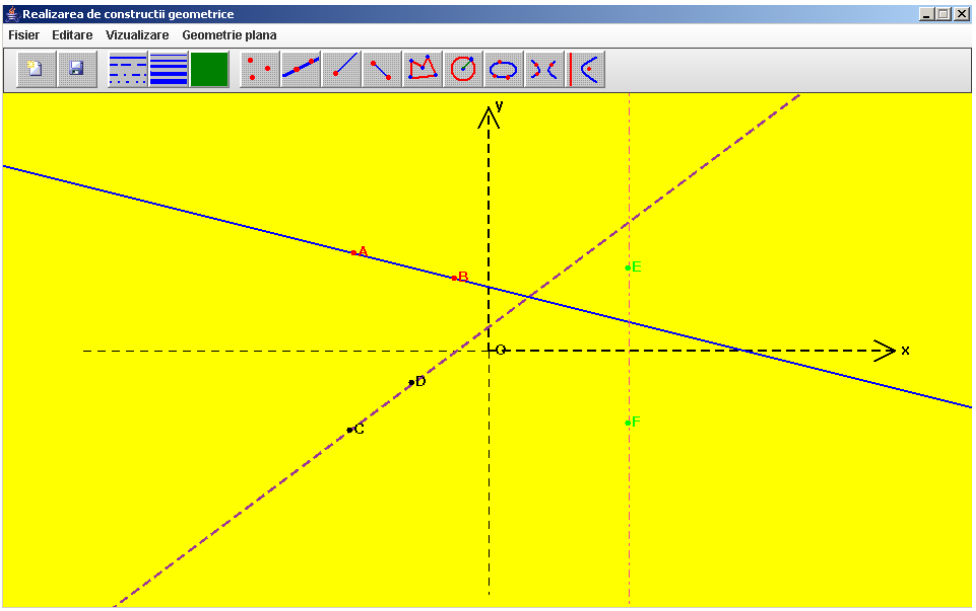


Figura 5.1.1. Drepte determinate de două puncte

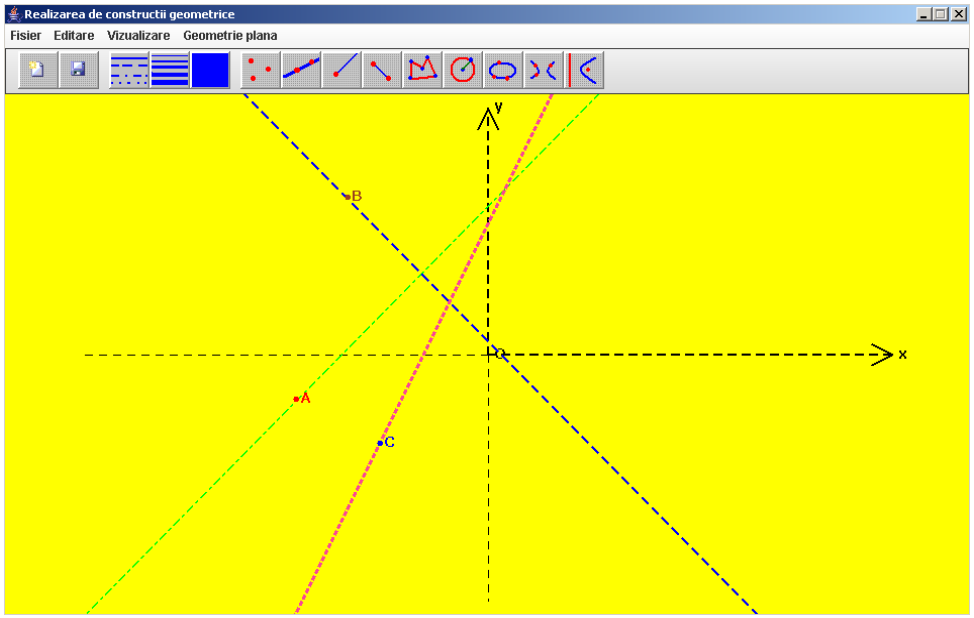


Figura 5.1.2. Drepte determinate de un punct

5.1.2 Operații cu puncte, drepte şi segmente

În acest subparagraf se urmăreşte prezentarea câtorva dintre cele mai importante opțiuni ale sistemului interactiv în care intervin puncte, drepte şi segmente, cum ar fi: mijlocul unui segment, intersecția a două drepte, paralela şi perpendiculara la o dreaptă printr-un punct, suportul unei semidrepte, suportul unui segment sau mediatoarea unui segment.

Intersecția a două drepte neparalele este un punct ale cărui coordonate se obțin rezolvând sistemul format din ecuațiile celor două drepte [92]:

$$\begin{cases} a_1 \cdot x + b_1 \cdot y + c_1 = 0 \\ a_2 \cdot x + b_2 \cdot y + c_2 = 0 \end{cases}, \text{ unde } a_1 \cdot b_2 \neq a_2 \cdot b_1 . \tag{5.1.3}$$

În cazul paralelei la o dreaptă printr-un punct specificat de coordonatele (x_0, y_0), iar dreapta dată are panta *m*, această nouă dreaptă va avea aceeaşi pantă [93], după cum se prezintă în continuare:

$$y - y_0 = m(x - x_0) \tag{5.1.4}$$

Dacă se doreşte determinarea perpendicularei la o dreaptă printr-un punct de coordonate (x_0, y_0), iar dreapta dată are panta *m*, atunci se poate utiliza ecuația anterioară în care se înlocuieşte panta cu $-1/m$.

În figura 5.1.3 sunt desenate perpendiculara şi paralela la o dreaptă, dar trec prin acelaşi punct.

Coordonatele mijlocului unui segment, ale cărui extremități au coordonatele (x_1, y_1) şi (x_2, y_2), se pot determina utilizând următoarele relații [92]:

$$x_M = \frac{x_1 + x_2}{2}, \; y_M = \frac{y_1 + y_2}{2}. \tag{5.1.5}$$

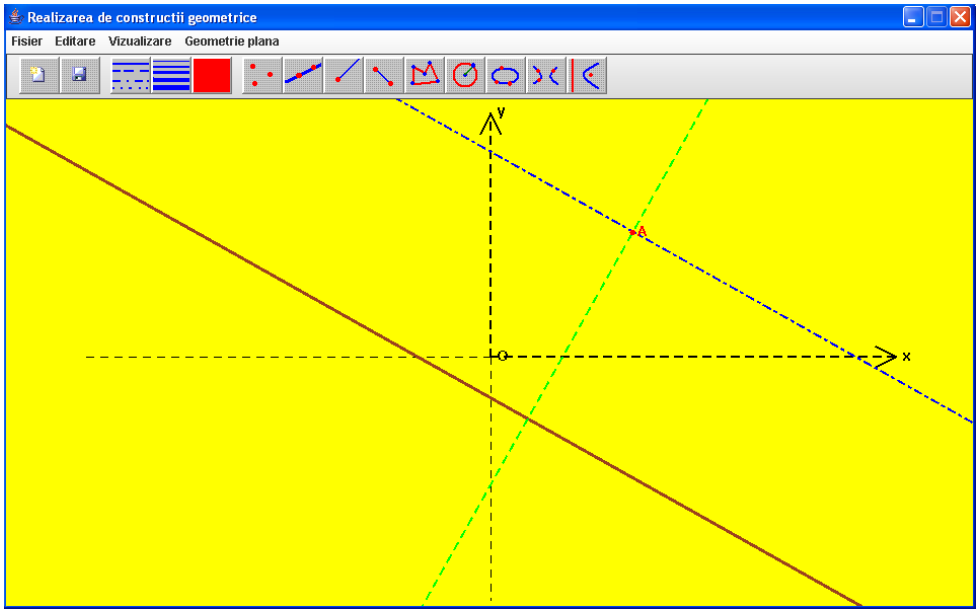


Figura 5.1.3. Perpendiculara şi paralela la o dreaptă

5.1.3 Reprezentarea poligoanelor

Toate tipurile de poligoane regulate sau neregulate, cum ar fi: triunghi, patrulater, hexagon, octogon, se pot desena prin specificarea vârfurilor [57, 59]. În cazul poligoanelor regulate este suficient să se specifice pe suprafaţa de desenare două vârfuri ale lor, deoarece toate laturile, respectiv unghiurile, poligoanelor regulate sunt egale [16]. În figura 5.1.4 sunt desenate patru poligoane neregulate: pentagon, heptagon, romb şi trapez, precum şi două poligoane regulate: hexagon şi octogon.

Sistemul dinamic permite desenarea tuturor tipurilor de poligoane în diverse modalităţi, iar pentru două cazuri particulare de poligoane: triunghi şi patrulater sunt prezentate atât rezulate teoretice, cât şi anumite tipuri de probleme rezolvate.

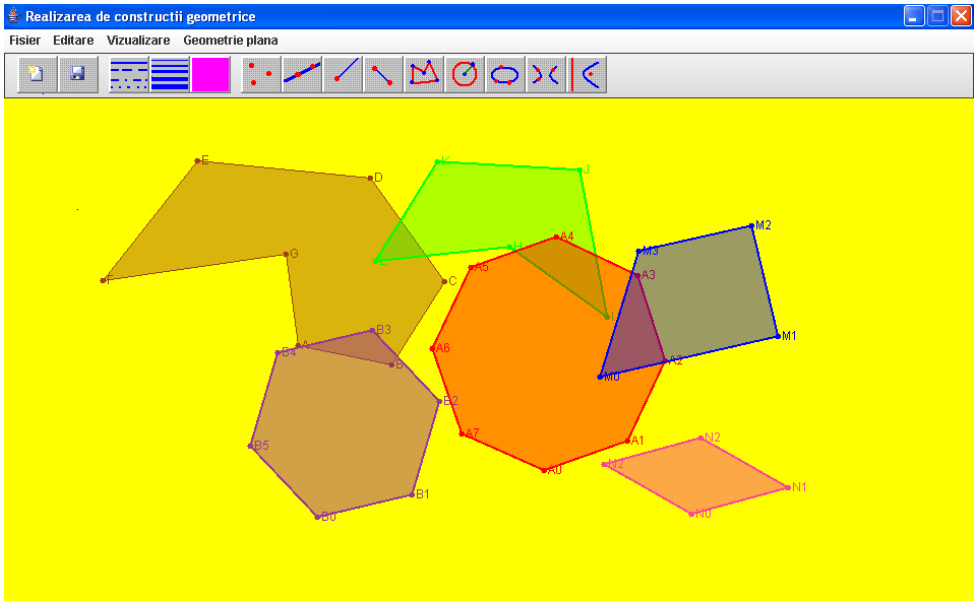


Figura 5.1.4. Realizarea poligoanelor

În al doilea caz, figurile geometrice corespunzătoare secvenței teoretice sunt dinamice. Prin selectarea unui vârf al triunghiului sau al patrulaterului şi modificarea coordonatelor acestuia, întreaga figură se va actualiza instantaneu.

5.1.4 Puncte importante într-un triunghi

În cazul prezentării noțiunilor teoretice şi rezultatelor referitoare la triunghi, sistemul dinamic cuprinde opțiuni corespunzătoare unor puncte caracteristice pentru triunghiuri, cum ar fi: centrul de greutate, ortocentrul, centrul cercului înscris (5.1.11), centrul cercului circumscris (fig. 5.1.7), centrul cercului lui Euler (fig. 5.1.7), punctul lui Lemoine (fig. 5.1.6), punctul lui Nagel, punctul lui Gergonne, punctul lui Vecten şi punctul lui Brocard. Aceste diverse reprezentări ale noțiunilor geometriei triunghiului au fost prezentate şi în câteva articole [53, 55]. Pentru desenarea acestor puncte s-au utilizat metode ale claselor: *Punct2D*, *Dreapta2D*, *Segment2D* şi *Triunghi2D*. Pentru orice punct selectat, sistemul determină şi afişează coordonatele punctului, dar şi caracteristicile teoretice ale acestuia.

Centrul de greutate, notat cu G, al unui triunghi este dat de intersecția medianelor, iar ortocentrul, notat cu H, este determinat de intersecția înălțimilor [76]. În figura 5.1.5 sunt reprezentate aceste două puncte atât pentru un triunghi ascuțitunghic, cât şi pentru unul obtuzunghic.

5.1.5 Linii importante într-un triunghi

În cazul prezentării noțiunilor teoretice şi rezultatelor referitoare la triunghi, sistemul dinamic cuprinde opțiuni corespunzătoare unor linii caracteristice pentru triunghiuri, cum ar fi: mediane (fig. 5.1.5), înălțimi (fig. 5.1.5), bisectoare, mediatoare, simediane, linie mijlocie, dreapta lui Euler, dreapta ortică, dreapta lui Simson sau dreapta lui Lemoine. Pentru desenarea acestor linii s-au utilizat metode ale claselor: *Dreapta2D*, *Segment2D* şi *Triunghi2D*. Pentru orice dreaptă selectată, sistemul determină şi afişează ecuația acesteia, dar şi caracteristicile teoretice ale sale.

Simediana unui triunghi corespunzătoare unui vârf este dreapta obținută prin simetrie față de bisectoarea interioară determinată de același vârf a medianei ce trece prin același vârf [76]. Cele trei simediane sunt concurente, punctul de intersecție fiind numit punctul lui Lemoine şi notat cu K. În figura 5.1.6 sunt reprezentate cele trei simediane împreună cu punctul de intersecție al lor atât pentru un triunghi ascuțitunghic, cât şi pentru unul obtuzunghic.

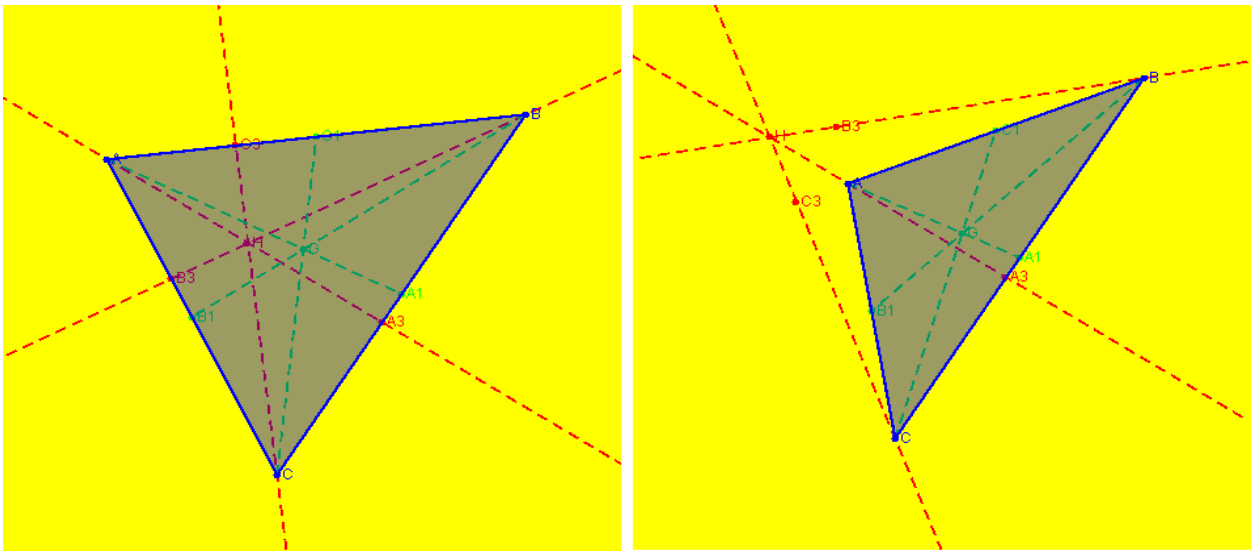


Figura 5.1.5. Centrul de greutate, ortocentrul, medianele şi înălțimile unui triunghi

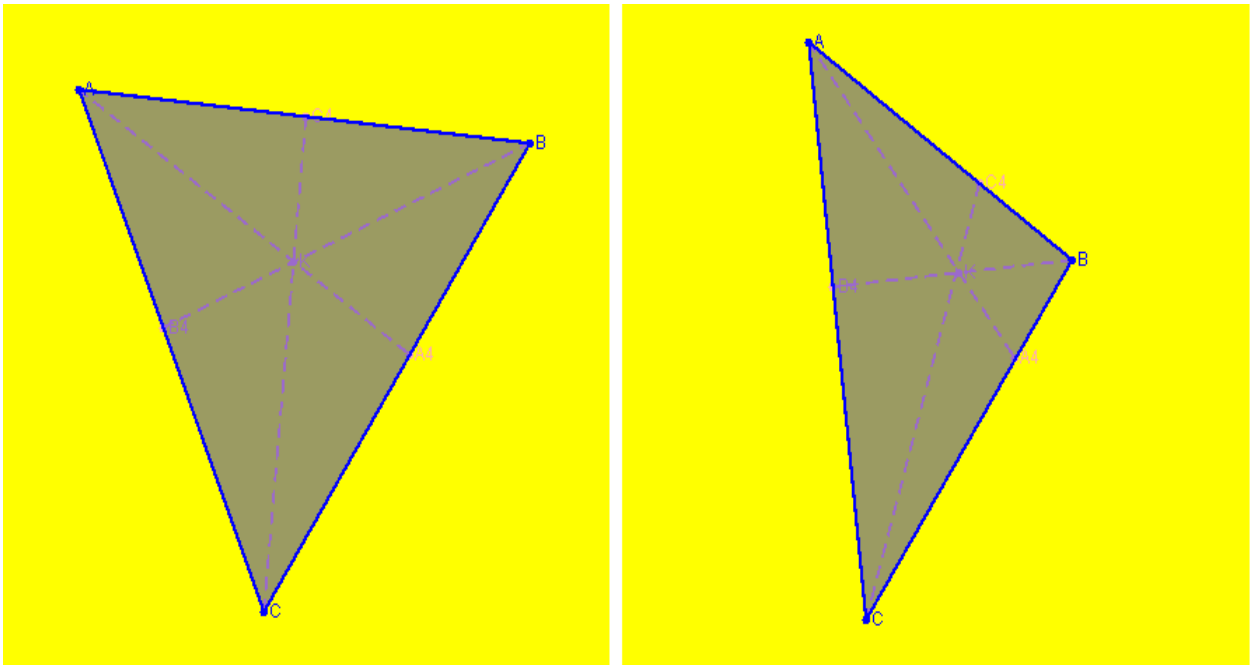


Figura 5.1.6. Punctul lui Lemoine şi simedianele unui triunghi

Punctul de intersecție al mediatoarelor unui triunghi, notat cu O, ortocentrul şi centrul de greutate sunt coliniare, dreapta determinată de aceste trei puncte numindu-se dreapta lui Euler [6]. În figura 5.1.7 este desenată această dreaptă împreună cu cele trei puncte care o determină atât pentru un triunghi ascuțitunghic, cât şi pentru unul obtuzunghic.

Tangentele la cercul circumscris unui triunghi neisocel în vârfurile lui taie laturile opuse în puncte situate pe o aceeaşi dreaptă, numită dreapta lui Lemoine [77]. În figura 5.1.8 este desenată această dreaptă împreună cu cercul circumscris pentru două triunghiuri obtuzunghice.

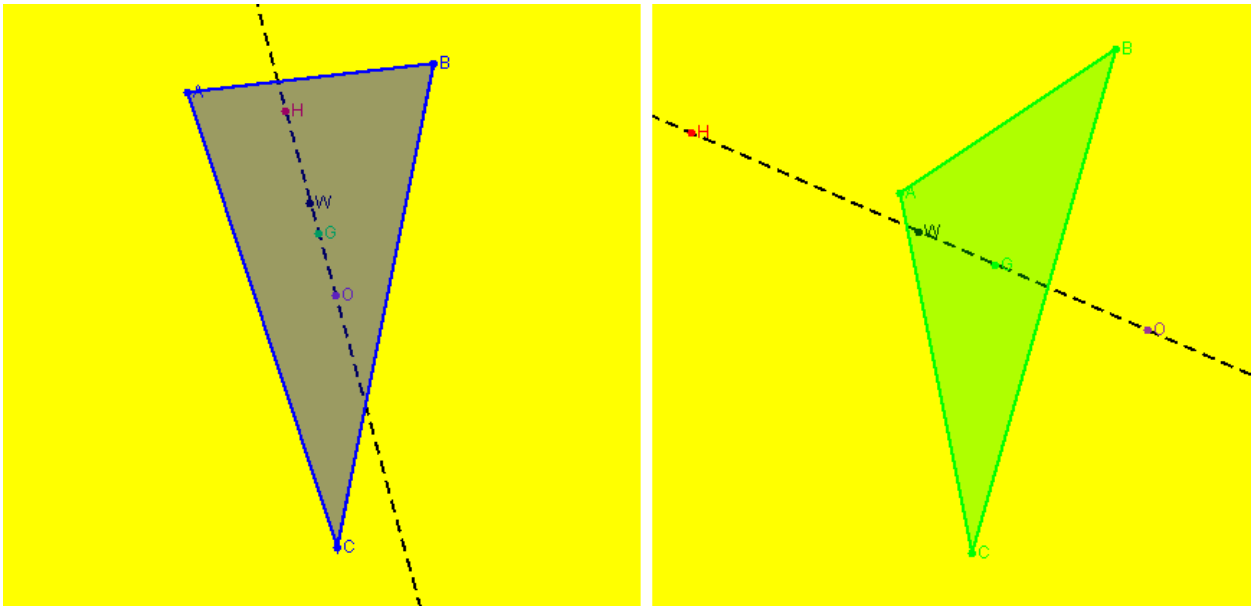


Figura 5.1.7. Dreapta lui Euler

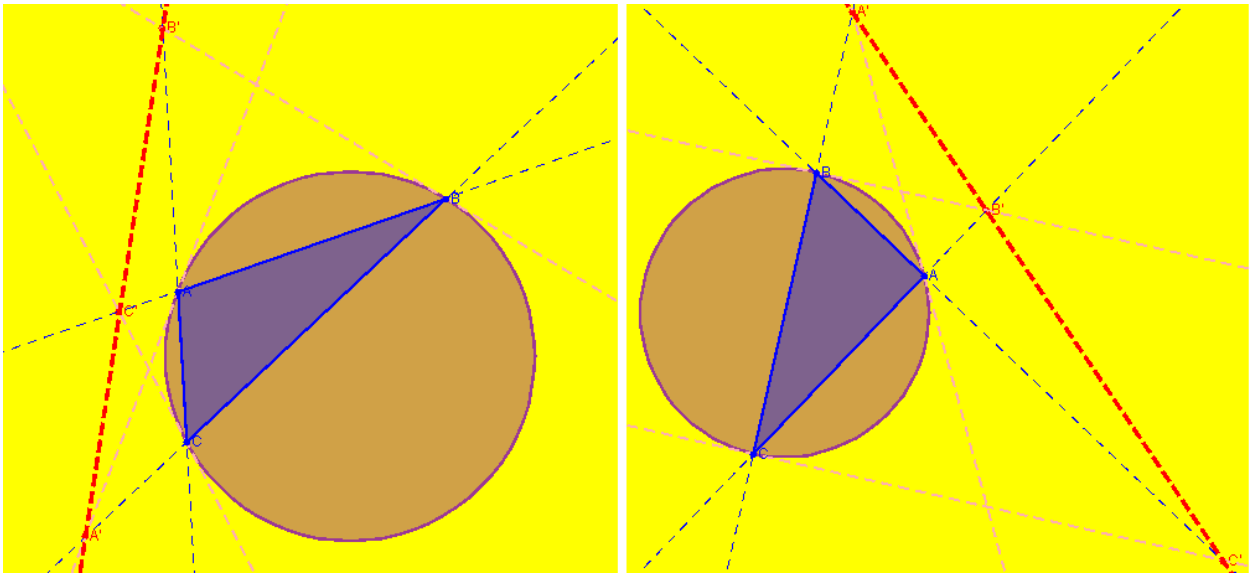


Figura 5.1.8. Dreapta lui Lemoine

5.1.6 Triunghiuri speciale pentru un triunghi

Sistemul interactiv cuprinde opțiuni corespunzătoare unor triunghiuri caracteristice pentru un triunghi, cum ar fi: triunghiul ortic şi triunghiul median. Pentru desenarea acestor triunghiuri s-au utilizat metode ale clasei *Triunghi2D*. Triunghiul ortic, reprezentat în figura 5.1.9, este determinat de picioarele celor trei înălțimi, iar triunghiul median, desenat în figura 5.1.10, este determinat de mijloacele celor trei laturi ale triunghiului inițial.

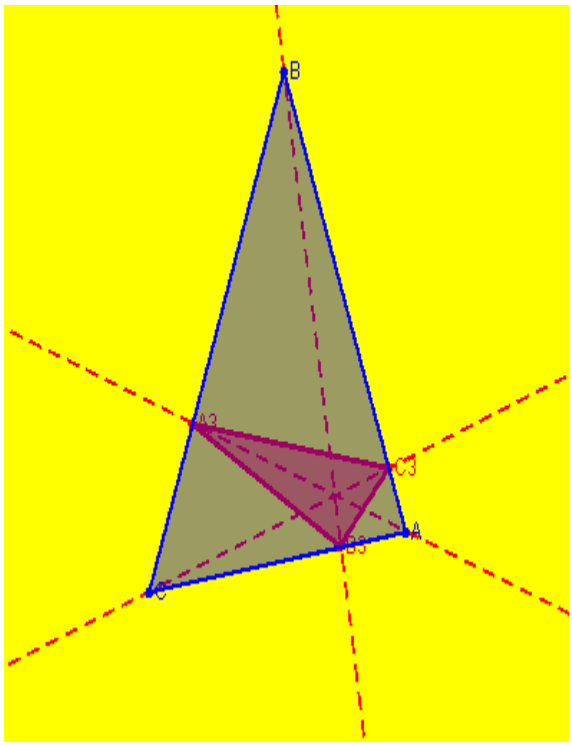


Figura 5.1.9. Triunghiul ortic

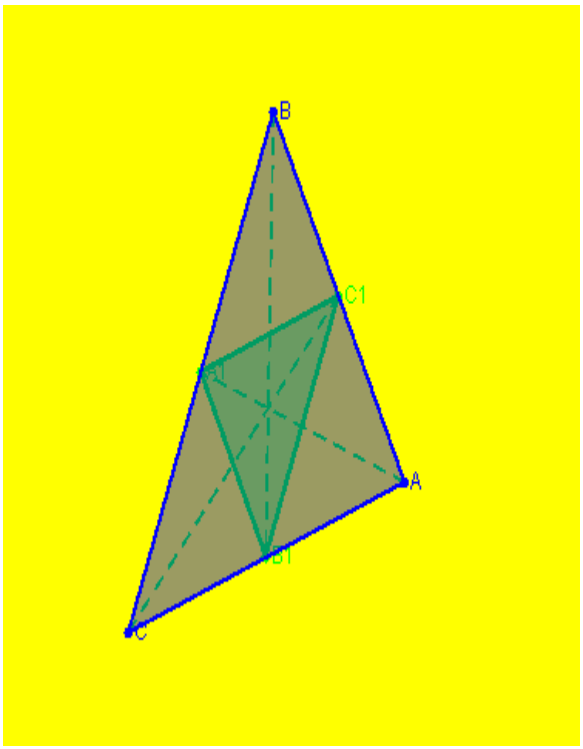


Figura 5.1.10. Triunghiul median

5.1.7 Cercuri caracteristice pentru un triunghi

În cazul prezentării noțiunilor teoretice şi rezultatelor referitoare la triunghi, sistemul dinamic cuprinde opțiuni corespunzătoare unor cercuri caracteristice pentru triunghiuri, cum ar fi: cercul înscris, cercul circumscris (fig. 5.1.8), cercuri exînscrise, cercul lui Euler (fig. 5.1.12), cercurile lui Lemoine (fig. 5.1.13) sau cercul lui Taylor (fig. 5.1.14). Aceste diverse reprezentări ale noțiunilor geometriei triunghiului au fost prezentate şi în câteva articole [53, 55]. Pentru desenarea acestor cercuri s-au utilizat metode ale claselor: *Punct2D*, *Triunghi2D* şi *Cerc2D*. Pentru orice cerc selectat, sistemul determină şi afişează coordonatele centrului cercului şi raza, dar şi caracteristicile teoretice ale acestuia.

Cercul înscris într-un triunghi are centrul determinat de intersecția bisectoarelor interioare, iar cercurile exînscrise au centrul în punctule intersecție a unei bisectoare interioare cu bisectoarele exterioare corespunzătoare celorlalte două vârfuri [6]. În figura 5.1.11 vor fi reprezentate aceste patru cercuri corespunzătoare unui triunghi ascuțitunghic, dar şi unui triunghi dreptunghic.

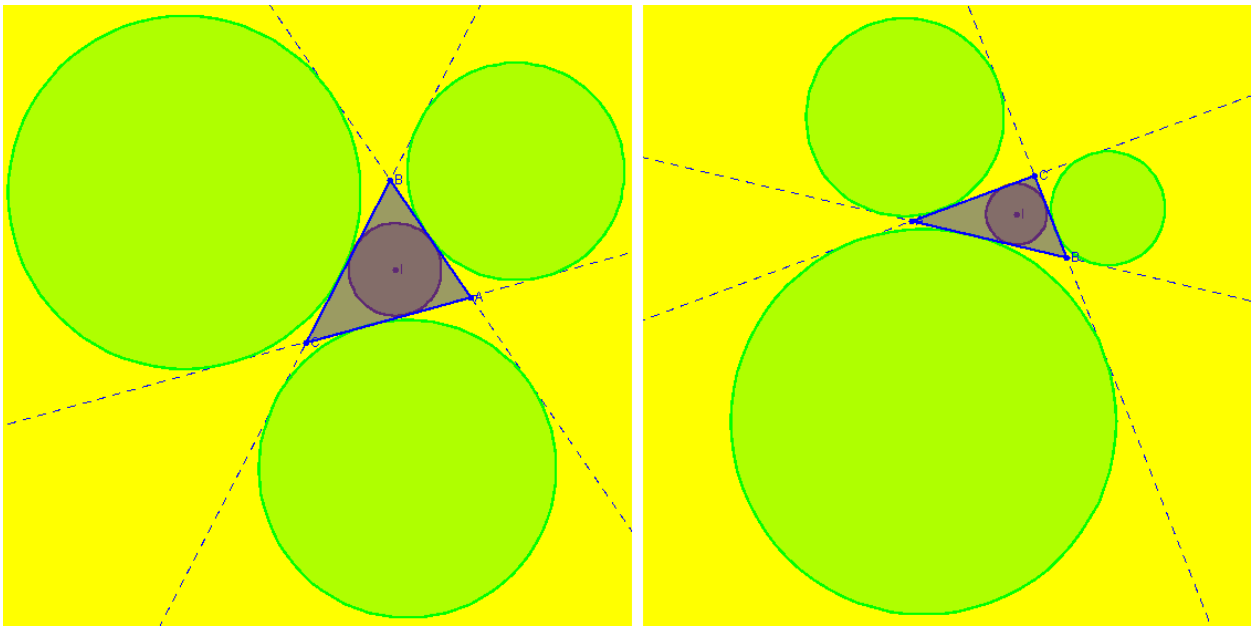


Figura 5.1.11. Cercul înscris şi cercurile exînscrise unui triunghi

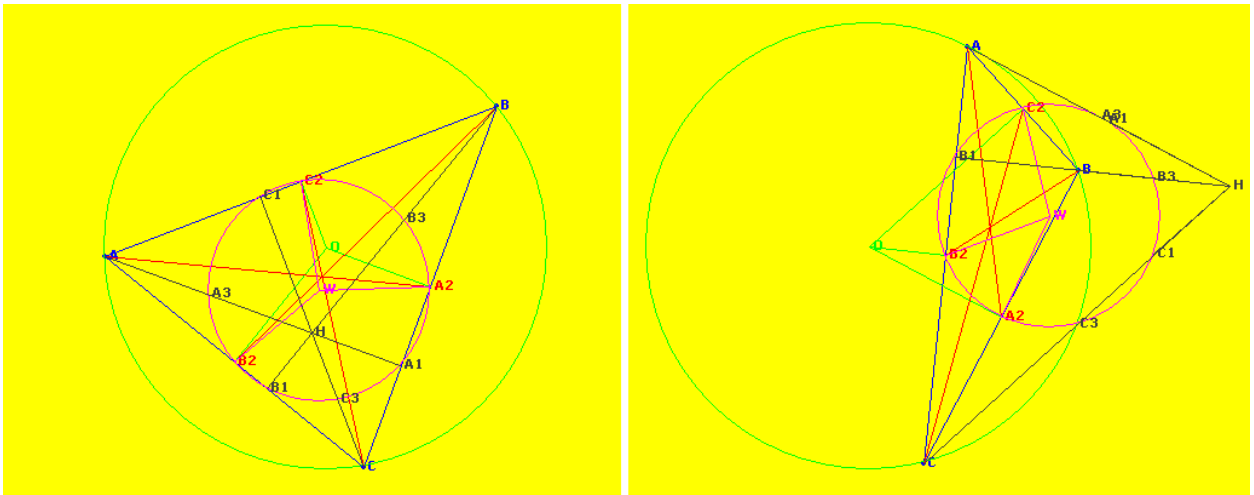


Figura 5.1.12. Cercul lui Euler

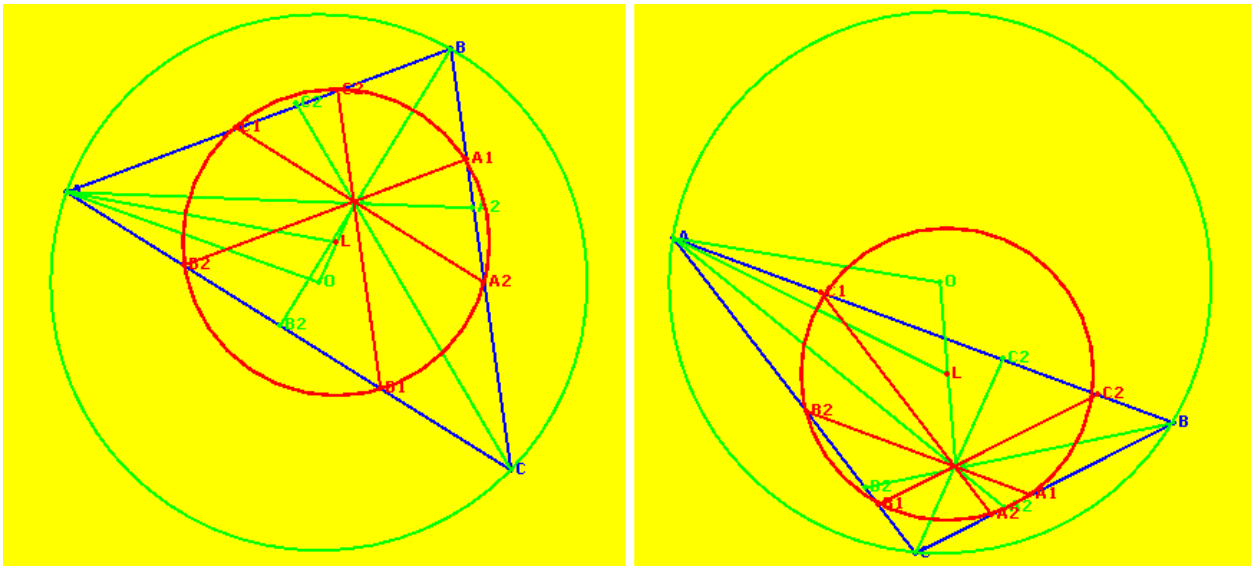


Figura 5.1.13. Primul cerc al lui Lemoine

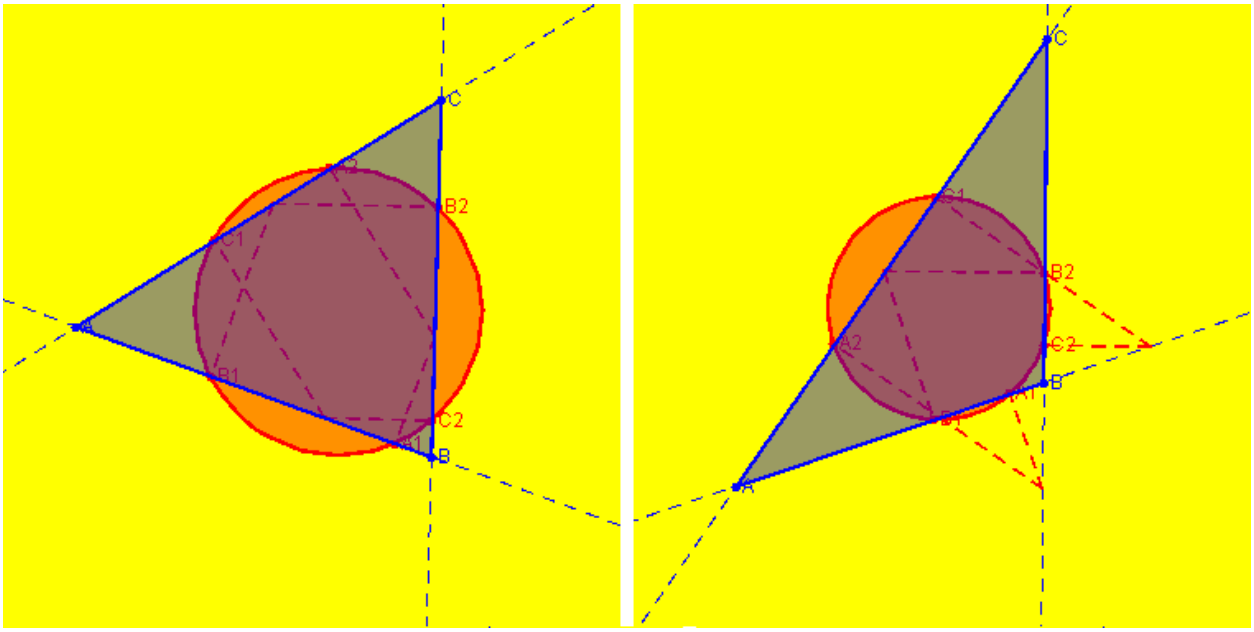


Figura 5.1.14. Cercul lui Taylor

5.1.8 Puncte importante într-un patrulater

În cazul prezentării noțiunilor teoretice şi rezultatelor referitoare la patrulater, sistemul dinamic cuprinde opțiuni corespunzătoare unor puncte caracteristice pentru patrulatere, cum ar fi: punctul lui Newton al unui patrulater circumscriptibil, punctul lui Miquel al unui patrulater convex (fig. 5.1.15) sau punctul lui Mathot al unui patrulater inscriptibil. Aceste diverse reprezentări ale noțiunilor geometriei patrulaterului au fost prezentate şi în câteva articole [55, 57, 59]. Pentru desenarea acestor puncte s-au utilizat metode ale claselor: *Punct2D*, *Dreapta2D*, *Segment2D* şi *Patrulater2D*. Pentru orice punct selectat, sistemul determină şi afişează coordonatele punctului, dar şi caracteristicile teoretice ale acestuia.

Într-un patrulater inscriptibil perpendicularele duse din mijloacele laturilor pe laturile opuse sunt concurente. Punctul de concurență se numeşte punctul lui Mathot [77]. În figura 5.1.16 este reprezentat acest punct în două cazuri.

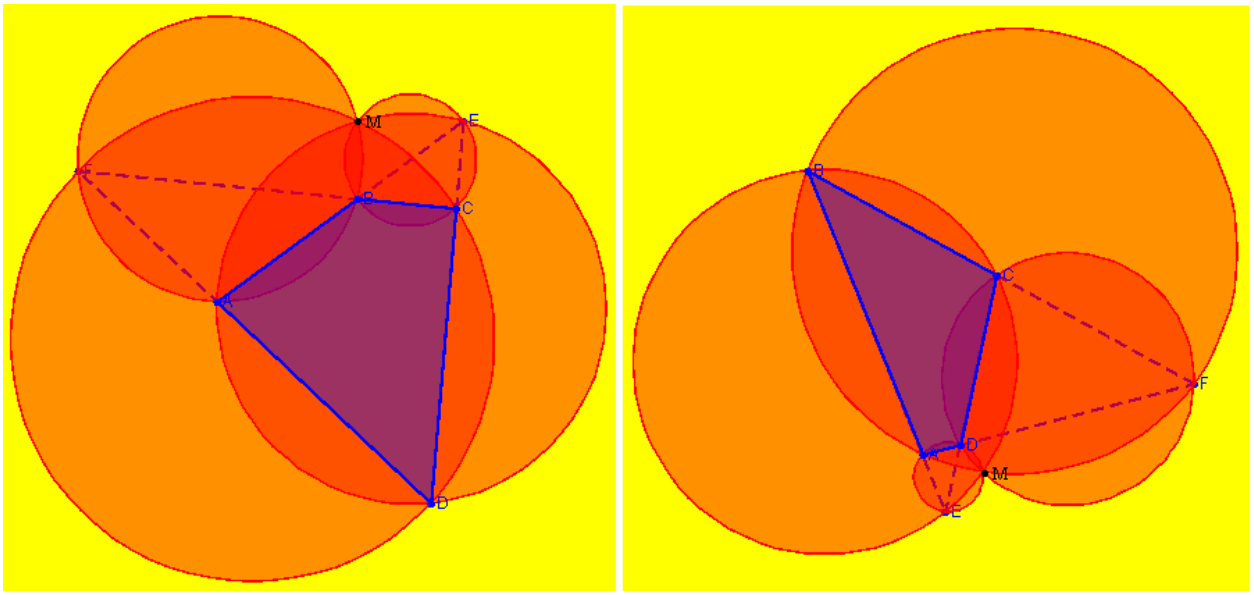


Figura 5.1.15. Punctul lui Miquel

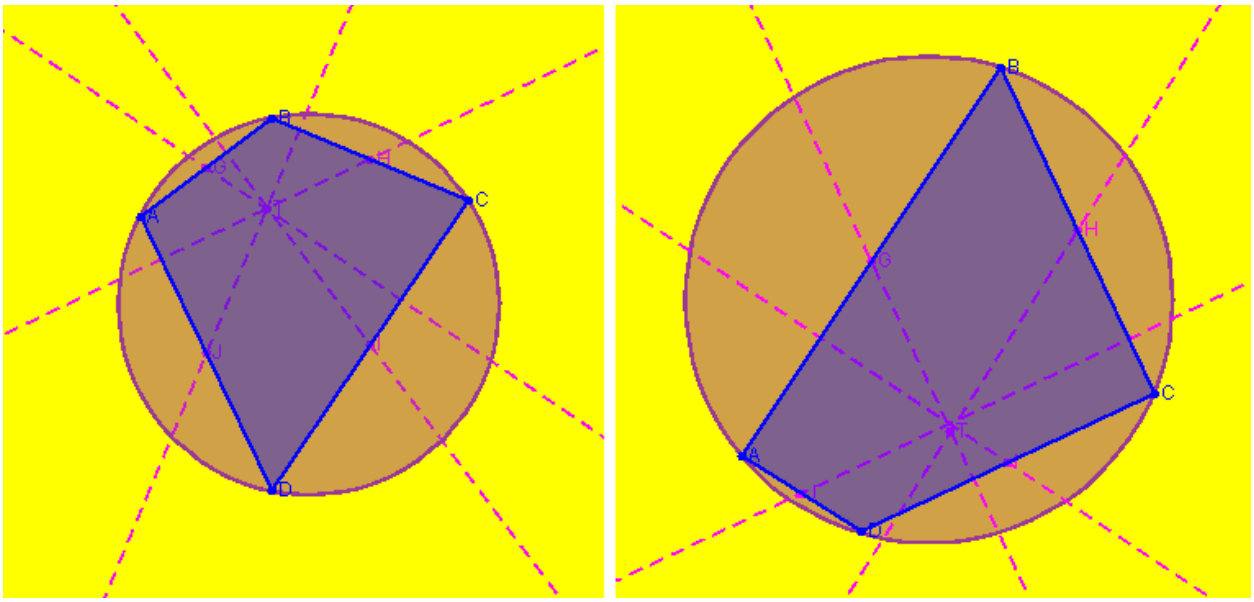


Figura 5.1.16. Punctul lui Mathot

5.1.9 Linii importante într-un patrulater

În cazul prezentării noțiunilor teoretice şi rezultatelor referitoare la patrulater, sistemul dinamic cuprinde opțiuni corespunzătoare unor linii caracteristice pentru patrulatere, cum ar fi: bimediane (fig. 5.1.17), diagonale (fig. 5.1.17), bisectoare, dreapta lui Newton corespunzătoare unui patrulater circumscriptibil (fig. 5.1.18), dreapta lui Gauss corespunzătoare unui patrulater complet sau dreapta lui Aubert corespunzătoare unui patrulater complet. Pentru desenarea acestor linii s-au utilizat metode ale claselor: *Dreapta2D*, *Segment2D* şi *Patrulater2D*. Pentru orice dreaptă selectată, sistemul determină şi afişează ecuația acesteia, dar şi caracteristicile teoretice ale sale.

Mijloacele diagonalelor unui patrulater complet sunt coliniare [77]. Dreapta determinată de aceste trei puncte se numeşte dreapta lui Gauss. În figura 5.1.19 este reprezentată această dreaptă împreună cu cele trei puncte care o determină atât pentru un patrulater convex, cât şi pentru un patrulater concav.

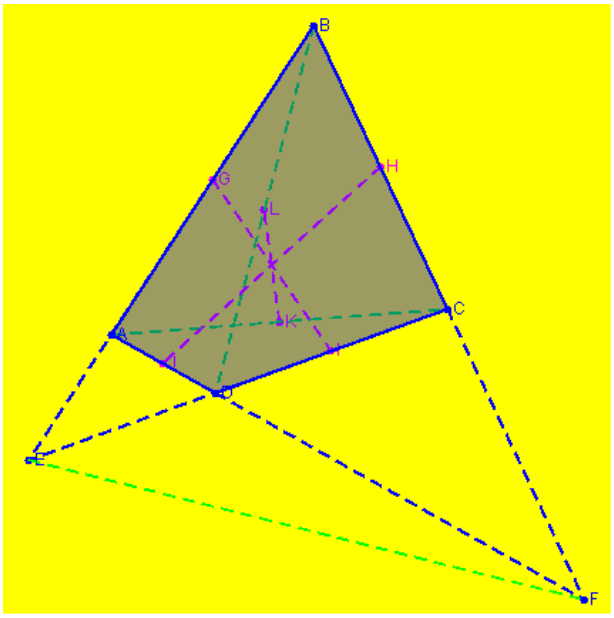


Figura 5.1.17. Diagonalele şi bimedianele patrulaterului

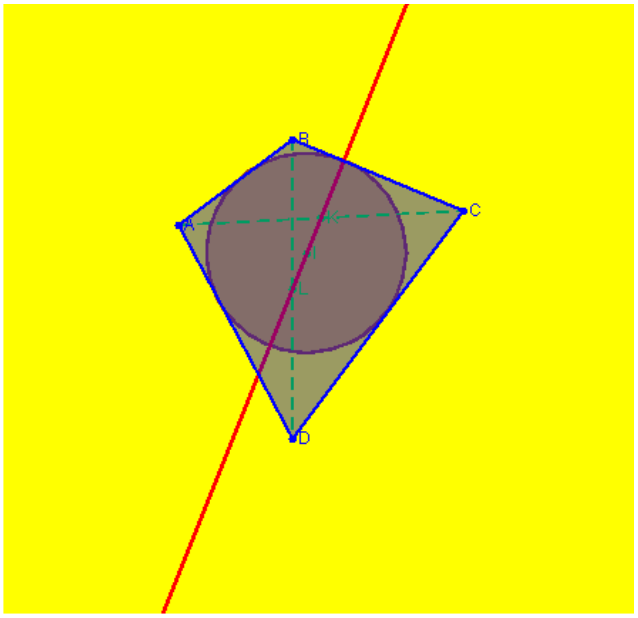
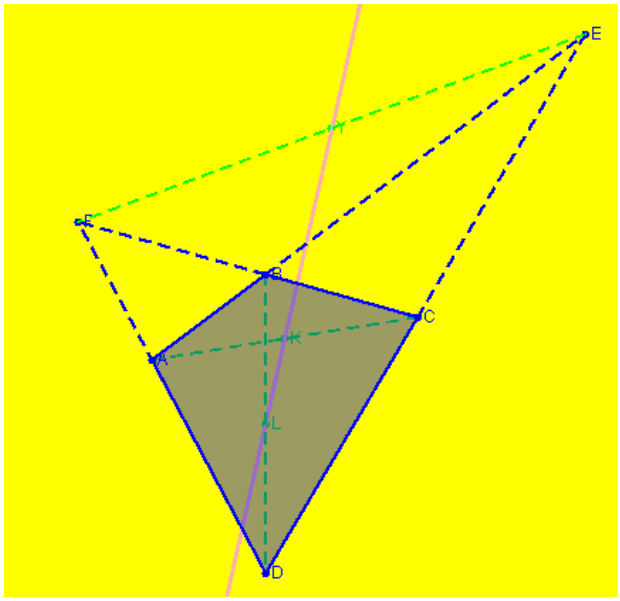


Figura 5.1.18. Dreapta lui Newton corespunzătoare unui patrulater circumscriptibil



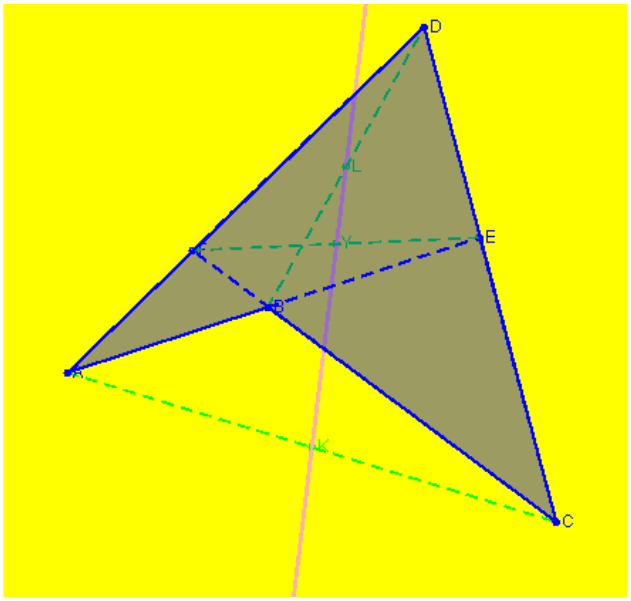


Figura 5.1.19. Dreapta lui Gauss

Ortocentrele celor patru triunghiuri formate cu laturile unui patrulater complet se găsesc pe aceeaşi dreaptă, numită dreapta lui Aubert [77]. În figura 5.1.20 este reprezentată această dreaptă împreună cu cele patru puncte care o determină atât pentru un patrulater convex, cât şi pentru un patrulater concav.

5.1.10 Cercuri caracteristice pentru un patrulater

În cazul prezentării noţiunilor teoretice şi rezultatelor referitoare la patrulater, sistemul dinamic cuprinde opţiuni corespunzătoare unor cercuri caracteristice pentru patrulatere, cum ar fi: cercul înscris într-un patrulater circumscriptibil (fig. 5.1.21) şi cercul circumscris unui patrulater inscriptibil (fig. 5.1.22). Pentru desenarea acestor cercuri s-au utilizat metode ale claselor: *Punct2D*, *Patrulater2D* şi *Cerc2D*. Pentru orice cerc selectat, sistemul determină şi afişează coordonatele centrului cercului şi raza, dar şi caracteristicile teoretice ale acestuia.

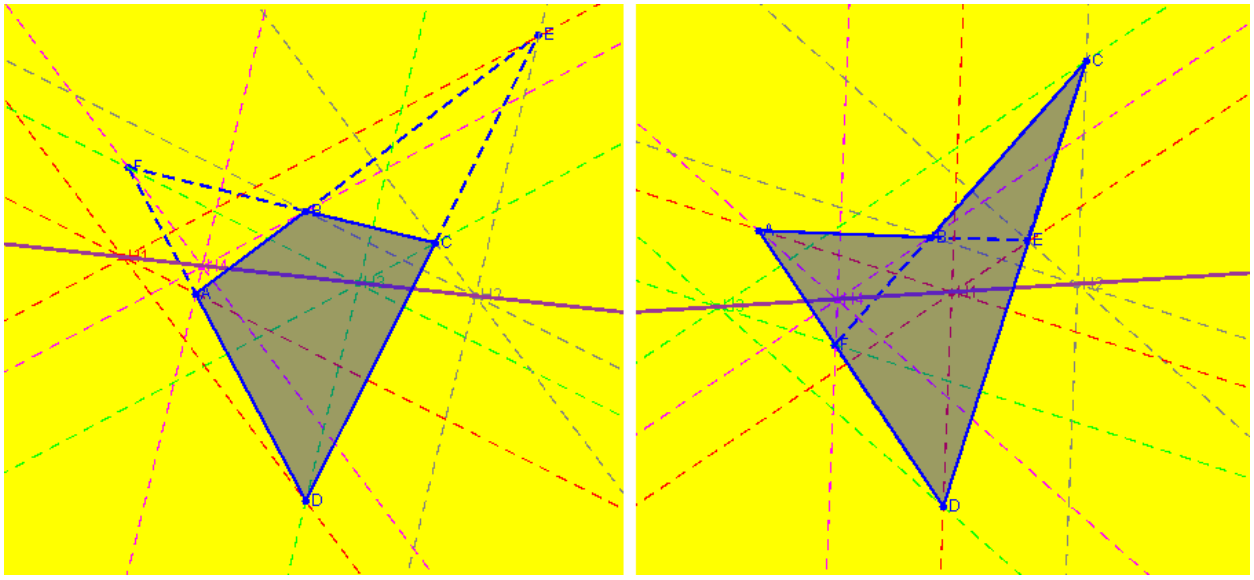


Figura 5.1.20. Dreapta lui Aubert

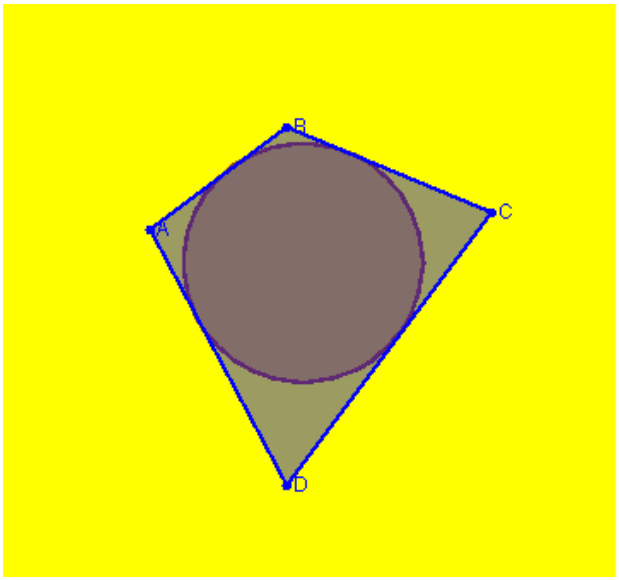


Figura 5.1.21. Cercul înscris într-un patrulater circumscriptibil

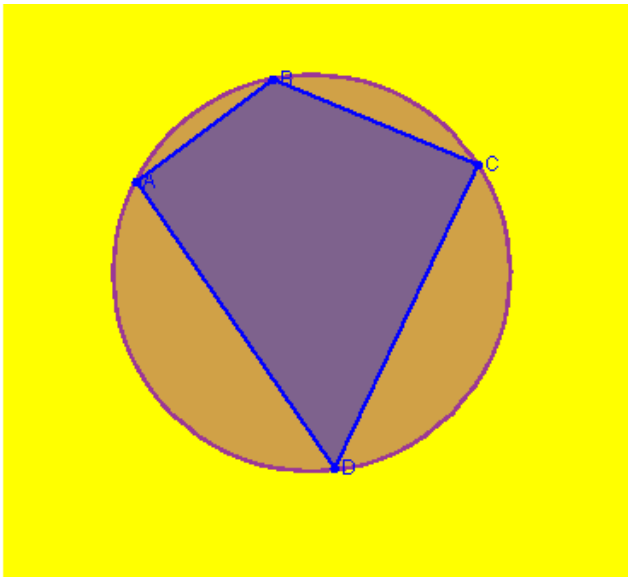


Figura 5.1.22. Cercul circumscris unui patrulater inscriptibil

5.2 Conice

5.2.1 Reprezentarea conicelor

Conicele sunt curbe de gradul al doilea ale planului euclidian bidimensional [109]. Ecuațiile conicelor se obțin prin anularea unui polinom de gradul al doilea în două variabile. Deci conicele sunt varietăți pătratice Γ^2 ale unui plan afin sau euclidian, dar de dimensiune 1. Ecuația generală a unei conice în sistemul de axe ortogonal (*Oxy*) al spațiului euclidian ε^2 este de forma:

$$\Gamma^2 : \ a_{11} \cdot x^2 + a_{22} \cdot y^2 + 2 \cdot a_{12} \cdot x \cdot y + 2 \cdot a_{10} \cdot x + 2 \cdot a_{20} \cdot y + a_{00} = 0, \qquad (5.2.1)$$

unde coeficienții a_{ij}, $i, j = \overline{1,3}$, sunt numere reale.

5.2.2 Conică determinată de cinci puncte

Pentru o manipulare interactivă, definiția unei conice prin intermediul matricii nu este foarte potrivită. Este mult mai preferabil a manipula conica în mod vizual prin conectare directă a unui punct pe ecran.

Cinci puncte definesc o conică în mod unic dacă patru dintre ele nu sunt coliniare [5]. Astfel poate fi creată o conică utilizând cinci puncte ca date de intrare şi având dată de ieşire conica determinată de aceste puncte. Ecuația conicei poate fi determinată scriind cinci ecuații ce conțin coordonatele cunoscute (x_i, y_i) ale celor cinci puncte aparținând conicei, obținând un sistem liniar de ecuații.

5.2.3 Modalități de determinare a unui cerc

Ecuația generală ce defineşte un cerc cu centrul în punctul de coordonate (x_0, y_0) şi de rază R este:

$$(x - x_0)^2 + (y - y_0)^2 = R^2. \tag{5.2.2}$$

Trei puncte definesc în mod unic un cerc dacă nu sunt coliniare [37]. Asftel, utilizând ca date de intrare cele trei puncte, poate fi instanțiat un obiect de tip *Cerc2D*. În figura 5.2.1 sunt reprezentate mai multe cercuri determinate de triplete de puncte necoliniare.

Altă modalitate de descriere a unui cerc este prin intermediul centrului şi razei. În figura 5.2.1 sunt reprezentate, de asemenea, şi cercuri determinate centru şi rază.

5.2.4 Modalități de determinare a unei elipse

Forma canonică a ecuației generale ce defineşte o elipsă cu centrul în punctul de coordonate (x_0, y_0) şi de parametrii *a* şi *b* este [7]:

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1. \tag{5.2.3}$$

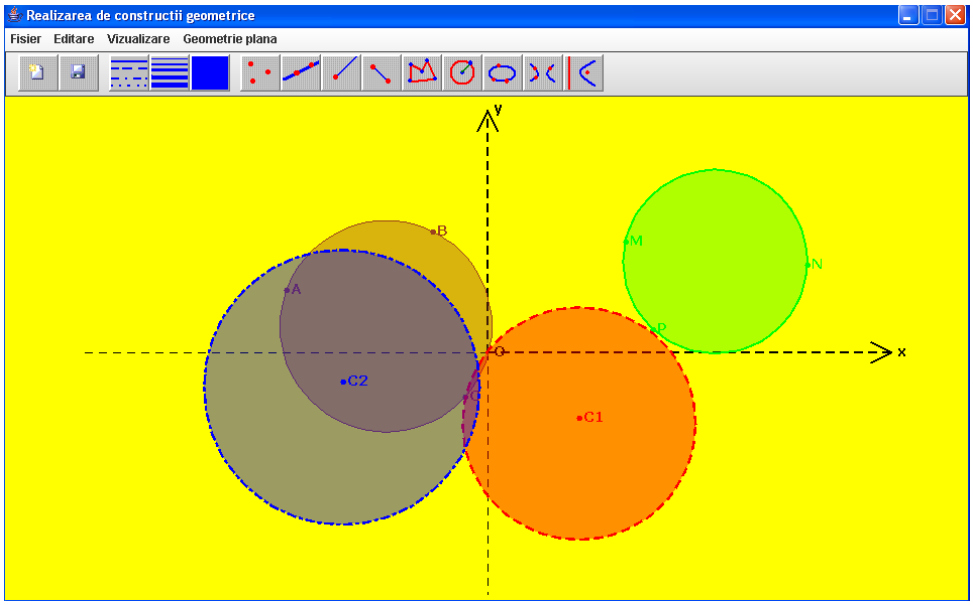


Figura 5.2.1. Cercuri determinate de trei puncte necoliniare, precum şi de centru şi rază

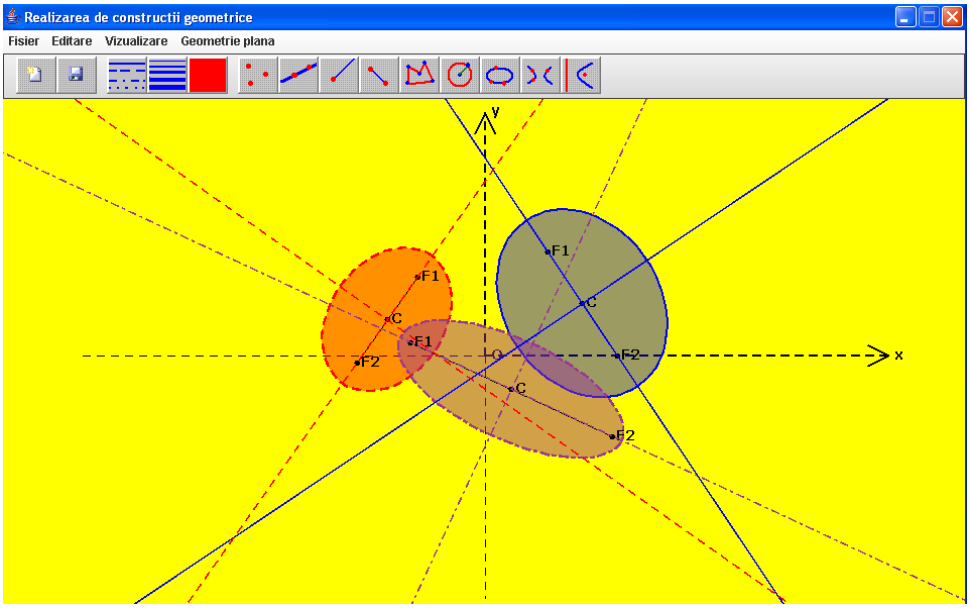


Figura 5.2.2. Elipse determinate de focare şi de parametrul *a*

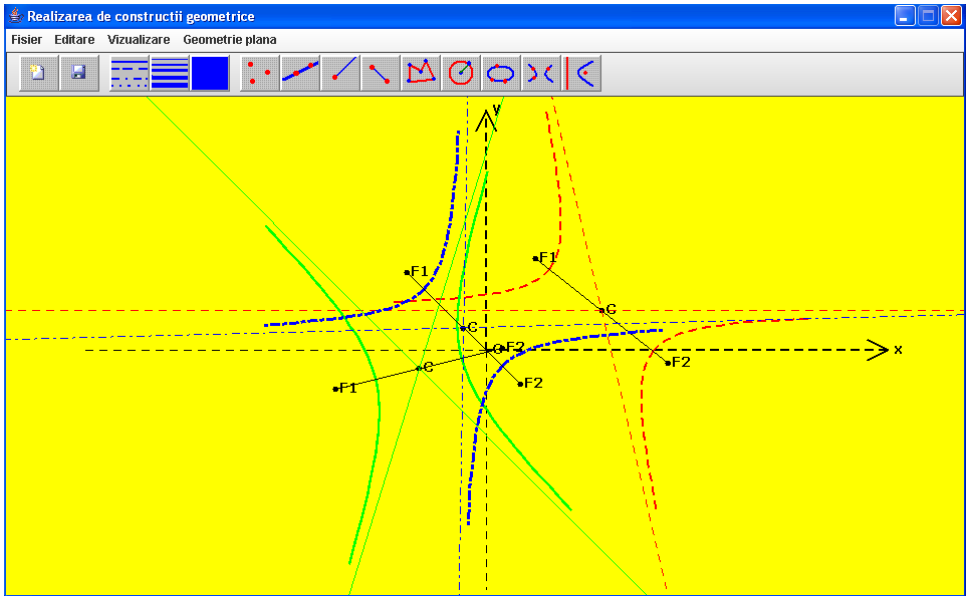


Figura 5.2.3. Hiperbole determinate de focare şi de parametrul *a*

O variantă de a defini o elipsă este prin specificarea celor două focare şi a unui parametru. În figura 5.2.2 sunt reprezentate elipse determinate de focare şi parametrul *a*, dar şi axele acestora.

5.2.5 Modalităţi de determinare a unei hiperbole

Forma canonică a ecuaţiei generale ce defineşte o hiperbolă cu centrul în punctul de coordonate (x_0, y_0) şi de parametrii *a* şi *b* este [89]:

$$\frac{(x-x_0)^2}{a^2} - \frac{(y-y_0)^2}{b^2} = 1. \qquad (5.2.4)$$

O variantă de a defini o hiperbolă este prin specificarea celor două focare şi a unui parametru. În figura 5.2.3 sunt reprezentate hiperbole determinate de focare şi parametrul *a*, dar

şi axele acestora. Aceste diverse reprezentări ale hiperbolelor, elipselor şi parabolelor au fost prezentate şi în câteva articole [56, 58].

5.2.6 Modalităţi de determinare a unei parabole

Forma canonică a ecuaţiei generale ce defineşte o parabolă cu vârful în punctul de coordonate (x_0, y_0) şi de parametru *p* este [90]:

$$y - y_0 = 2 \cdot p \cdot (x - x_0). \tag{5.2.5}$$

O variantă de a defini o hiperbolă este prin specificarea focarului şi a dreptei directoare. În figura 5.2.4 sunt reprezentate parabole determinate de focar şi dreapta directoare.

5.2.7 Tangenta şi normala într-un punct la elipsă

Ecuaţia tangentei într-un punct M dat la o elipsă, pentru care axa focarelor este paralelă cu axa Ox, se obţine prin procedeul de dedublare:

$$\frac{x_1 - x_0}{a^2} \cdot x + \frac{y_1 - y_0}{b^2} \cdot y + \frac{x_0 \cdot (x_0 - x_1)}{a^2} + \frac{y_0 \cdot (y_0 - y_1)}{b^2} = 0, \text{ unde } C(x_0 y_0), M x_1, y_1). \tag{5.2.5}$$

Ecuaţia normalei se obţine pornind de la proprietatea că tangenta şi normala în acelaşi punct la o curbă sunt perpendiculare. În figura 5.2.5 sunt prezentate tangenta şi normala la elipsă într-un punct specificat, iar în figura 5.2.6 sunt prezentate tangenta şi normala la cerc într-un punct specificat. Pentru desenarea acestor drepte s-au utilizat metode ale claselor: *Dreapta2D, Cerc2D* şi *Elipsa2D*.

5.2.8 Tangenta şi normala într-un punct la hiperbolă

Ecuaţia tangentei într-un punct M dat la o hiperbolă, pentru care axa focarelor este paralelă cu axa Ox, se obţine prin procedeul de dedublare:

$$\frac{x_1 - x_0}{a^2} \cdot x - \frac{y_1 - y_0}{b^2} \cdot y - \frac{x_0 \cdot (x_0 - x_1)}{a^2} + \frac{y_0 \cdot (y_0 - y_1)}{b^2} = 0, \text{ unde } C(x_0 y_0), M x_1, y_1). \tag{5.2.6}$$

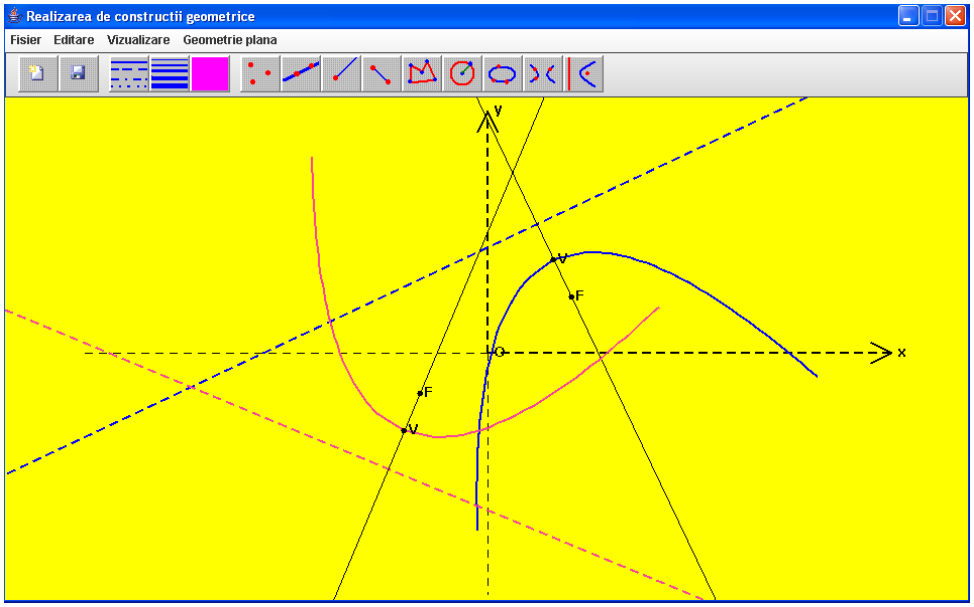


Figura 5.2.4. Parabole determinate de focar şi de dreapta directoare

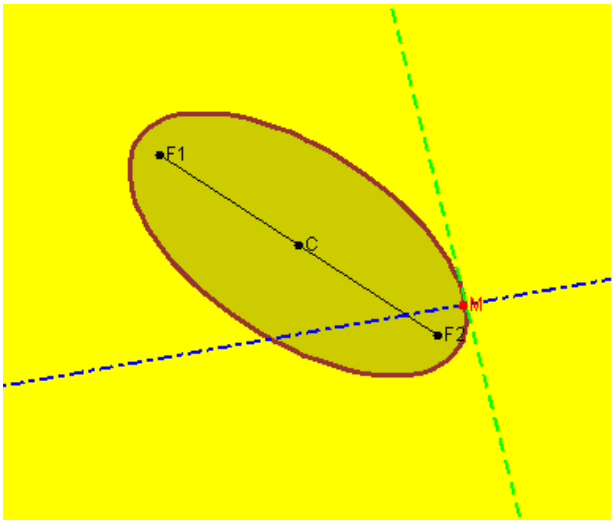


Figura 5.2.5. Tangenta şi normala într-un punct la elipsă

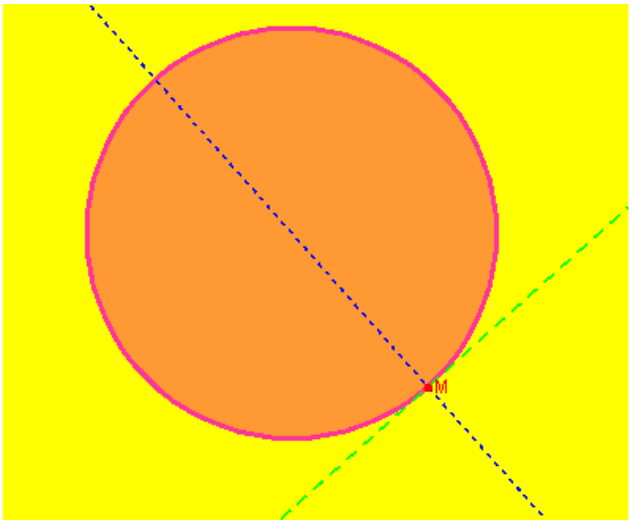


Figura 5.2.6. Tangenta şi normala într-un punct la cerc

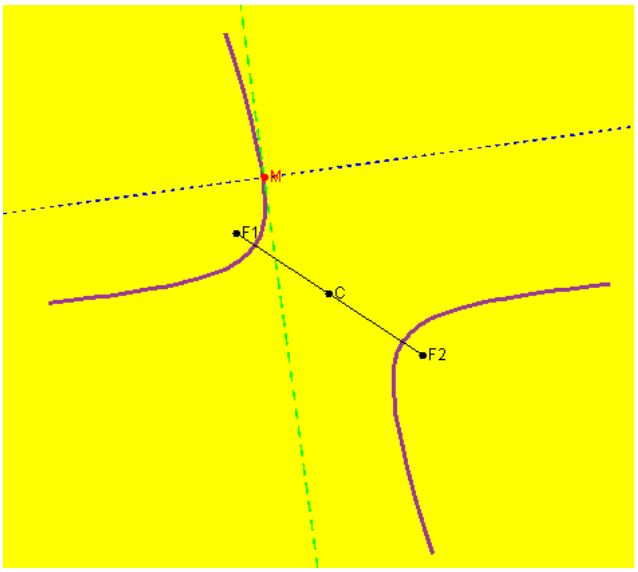


Figura 5.2.7. Tangenta şi normala într-un punct la hiperbolă

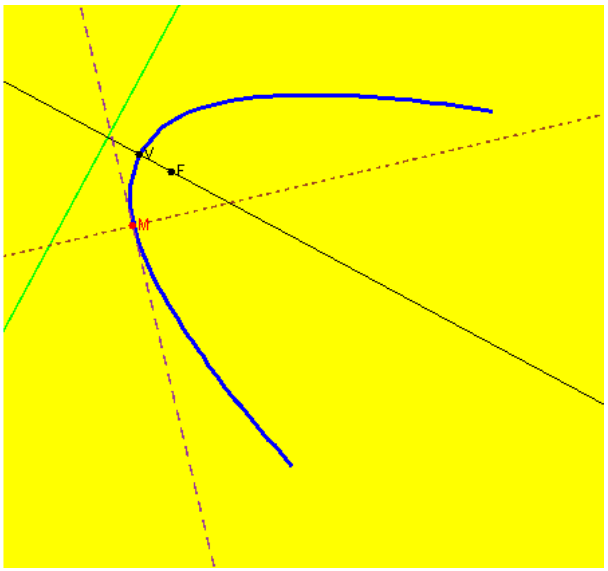


Figura 5.2.8. Tangenta şi normala într-un punct la parabolă

Ecuația normalei se obține pornind de la proprietatea că tangenta şi normala în acelaşi punct la o curbă sunt perpendiculare. În figura 5.2.7 sunt prezentate tangenta şi normala la hiperbolă într-un punct specificat. Pentru desenarea acestor drepte s-au utilizat metode ale claselor: *Dreapta2D* şi *Hiperbola2D*.

5.2.9 Tangenta şi normala într-un punct la parabolă

Ecuația tangentei într-un punct M dat la o parabolă, pentru care axa determinată de vârf şi focar este paralelă cu axa Ox, se obține prin procedeul de dedublare:

$$-p\cdot x+(y_1-y_0)\cdot y-p\cdot x_1-y_0\cdot(y_1-y_0)=0\text{, unde } V(x_0y_0)\text{, M}\,x_1,y_1). \qquad (5.2.7)$$

Ecuația normalei se obține pornind de la proprietatea că tangenta şi normala în acelaşi punct la o curbă sunt perpendiculare. În figura 5.2.8 sunt prezentate tangenta şi normala la parabolă într-un punct specificat. Pentru desenarea acestor drepte s-au utilizat metode ale claselor: *Dreapta2D* şi *Parabola2D*.

5.2.10 Intersecția unui cerc cu o dreaptă

O altă operație de bază a sistemului interactiv este aceea de a determina intersecția dintre un cerc şi o dreaptă. Rezultatul acestei intersecții poate fi un punct (în caz de tangență), două puncte sau mulțimea vidă [107]. Datele de ieşire pentru această opțiune se determină rezolvând următorul sistem, unde prima ecuație identifică dreapta, iar a doua este ecuația cercului de centru (x_0, y_0) şi de rază R:

$$\begin{cases} a \cdot x + b \cdot y + c = 0 \\ (x - x_0)^2 + (y - y_0)^2 = R^2 \end{cases}. \tag{5.2.8}$$

În figura 5.2.9 sunt prezentate intersecții între perechi de drepte şi cercuri, în primul caz datele de ieşire fiind două puncte, iar în al doilea caz rezultatul este un punct. Pentru determinarea acestor intersecții s-au utilizat metode ale claselor: *Dreapta2D* şi *Cerc2D*.

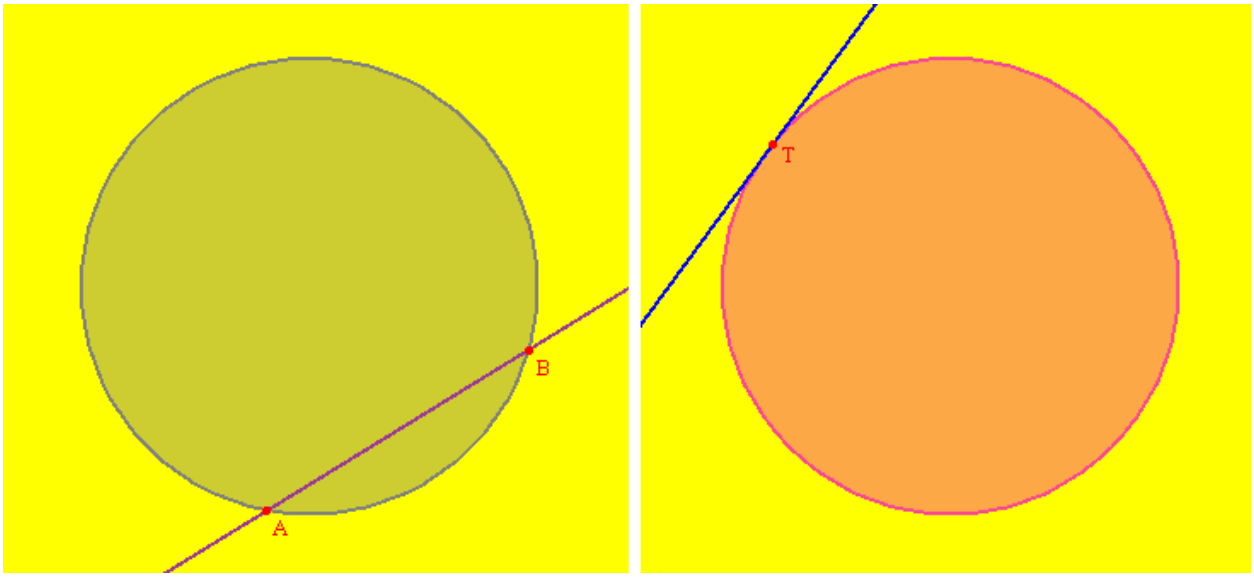


Figura 5.2.9. Intersecția unui cerc cu o dreaptă

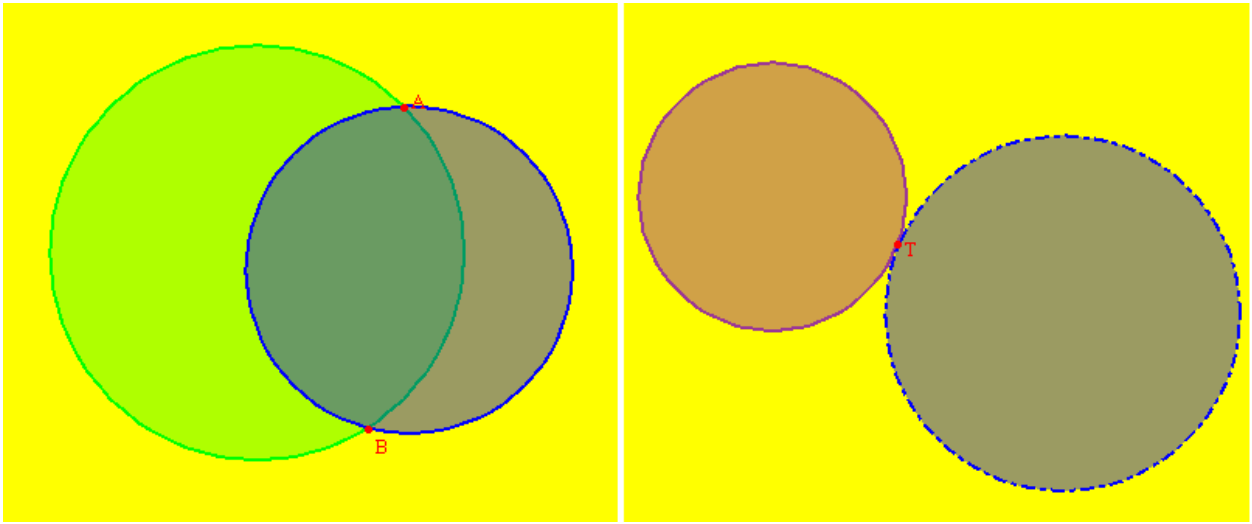


Figura 5.2.10. Intersecția a două cercuri

5.2.11 Intersecția dintre două cercuri

Rezultatul acestei operații poate fi un punct (în caz de tangență), două puncte sau mulțimea vidă [107]. Datele de ieşire pentru această opțiune se determină rezolvând următorul sistem, unde prima ecuație identifică cercul de centru (x_1, y_1) şi de rază R_1, iar a doua ecuație identifică cercul de centru (x_2, y_2) şi de rază R_2:

$$\begin{cases}(x-x_1)^2+(y-y_1)^2=R_1^2\\(x-x_2)^2+(y-y_2)^2=R_2^2\end{cases}. \tag{5.2.9}$$

În figura 5.2.10 sunt prezentate intersecții între perechi de cercuri, în primul caz datele de ieşire fiind două puncte, iar în al doilea caz rezultatul este un punct. Pentru determinarea acestor intersecții s-au utilizat metode ale clasei *Cerc2D*.

5.3 Cuadrice

Sistemul dinamic permite desenarea mai multor tipuri de cuadrice în diverse modalități, dar prezintă atât rezulate teoretice, cât şi anumite tipuri de probleme rezolvate. Aceste diverse reprezentări ale cuadricelor au fost prezentate într-un articol [54].

5.3.1 Modalități de determinare a unui elipsoid

Ecuația generală ce defineşte un elipsoid cu centrul în punctul de coordonate (x_0, y_0, z_0) şi de parametrii *a, b* şi *c* este [110]:

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}+\frac{(z-z_0)^2}{c^2}=1. \tag{5.3.1}$$

Ecuațiile parametrice, utilizate în reprezentarea elipsoidului, sunt:

$$\begin{cases}x=x_0+a\cdot cos\,u\cdot cos\,v\\y=y_0+b\cdot cos\,u\cdot sin\,v\\z=z_0+c\cdot sin\,u\end{cases}, \text{ unde } u\in\left[-\frac{\pi}{2},\frac{\pi}{2}\right],\ v\in[0,2\pi]. \tag{5.3.2}$$

O variantă de a defini un elipsoid este prin specificarea celor trei parametrii şi a coordonatelor centrului. În figura 5.3.1 sunt reprezentate mai multe astfel de cuadrice prin utilizarea următoarelor două proiecții: $(\alpha=-145, \beta=-30, \gamma=90)$ şi $(\alpha=-120, \beta=30, \gamma=145)$.

5.3.2 Modalități de determinare a unui hiperboloid

Ecuația generală ce defineşte un hiperboloid cu o pânză cu centrul în punctul de coordonate (x_0, y_0, z_0) şi de parametrii *a, b* şi *c* este [110]:

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}-\frac{(z-z_0)^2}{c^2}=1. \tag{5.3.3}$$

Ecuația generală ce defineşte un hiperboloid cu două pânze cu centrul în punctul de coordonate (x_0, y_0, z_0) şi de parametrii *a, b* şi *c* este [110]:

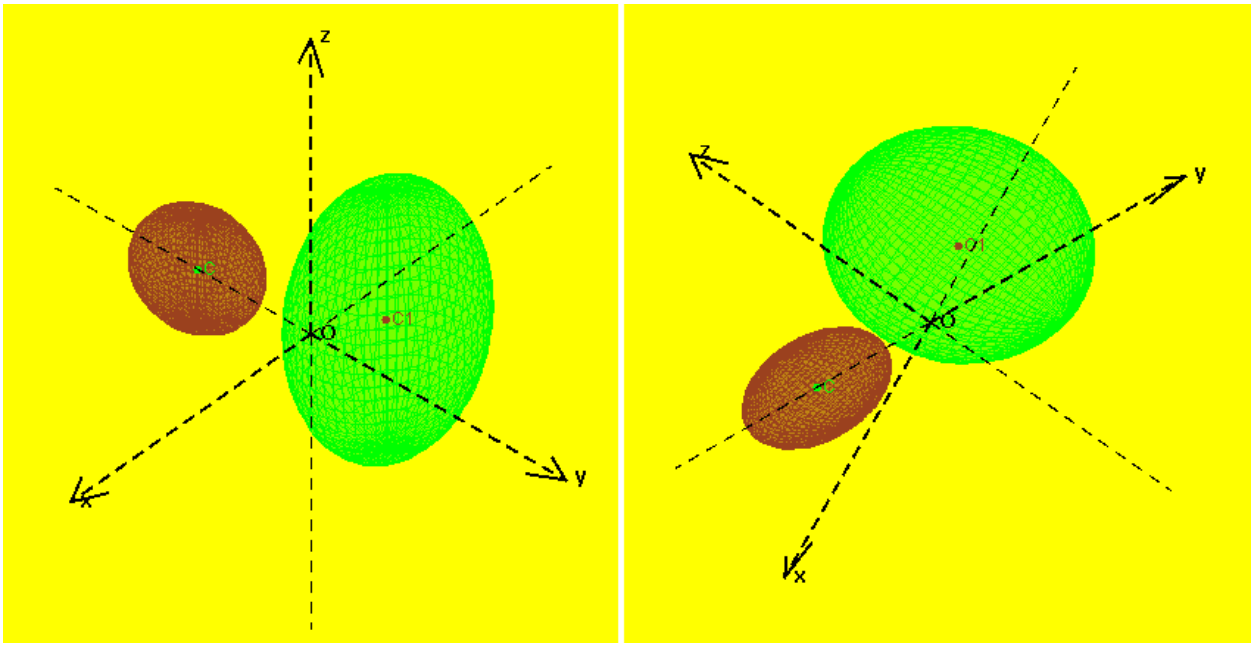


Figura 5.3.1. Desenarea elipsoidului

$$\frac{(x-x_0)^2}{a^2}-\frac{(y-y_0)^2}{b^2}-\frac{(z-z_0)^2}{c^2}=1. \qquad (5.3.4)$$

O variantă de a defini un hiperboloid este prin specificarea celor trei parametrii şi a coordonatelor centrului. În figura 5.3.2 sunt reprezentate mai multe astfel de cuadrice prin utilizarea următoarelor două proiecţii: $(\alpha=-150,\beta=-45,\gamma=90)$ şi $(\alpha=-120,\beta=50,\gamma=145)$.

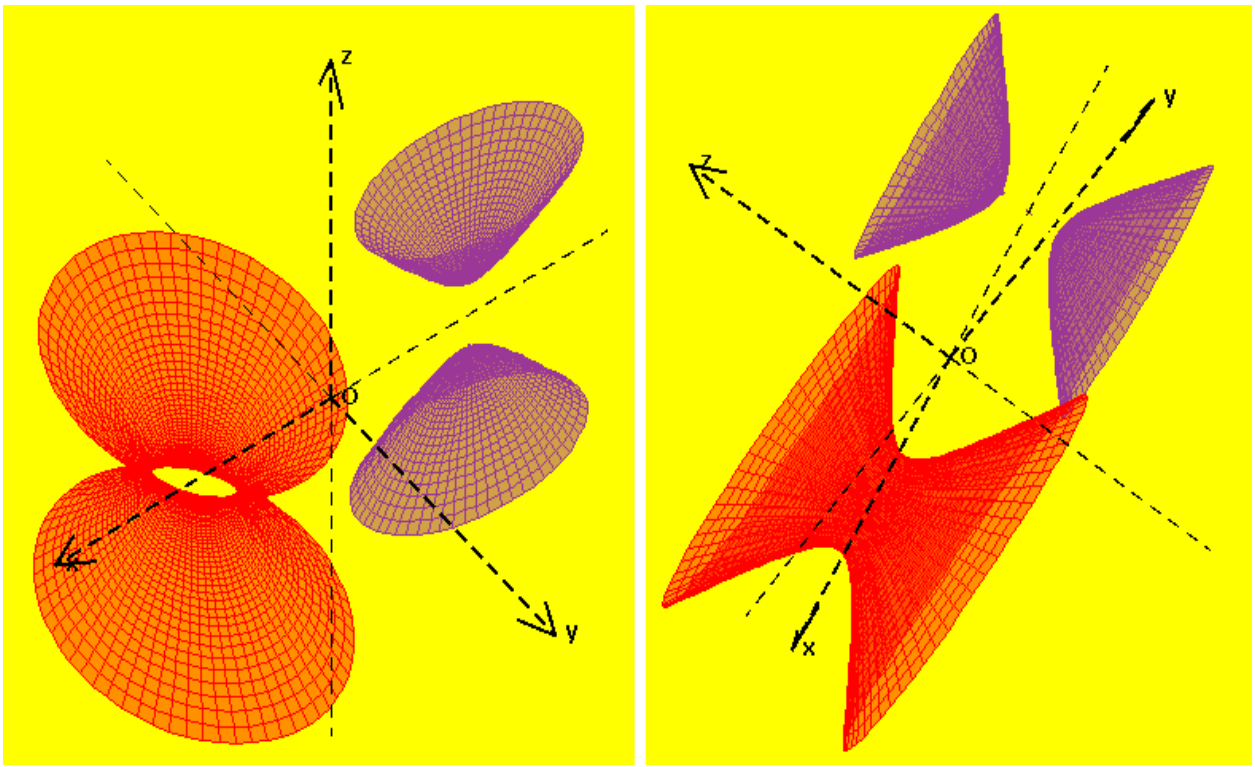


Figura 5.3.2. Desenarea hiperboloidului

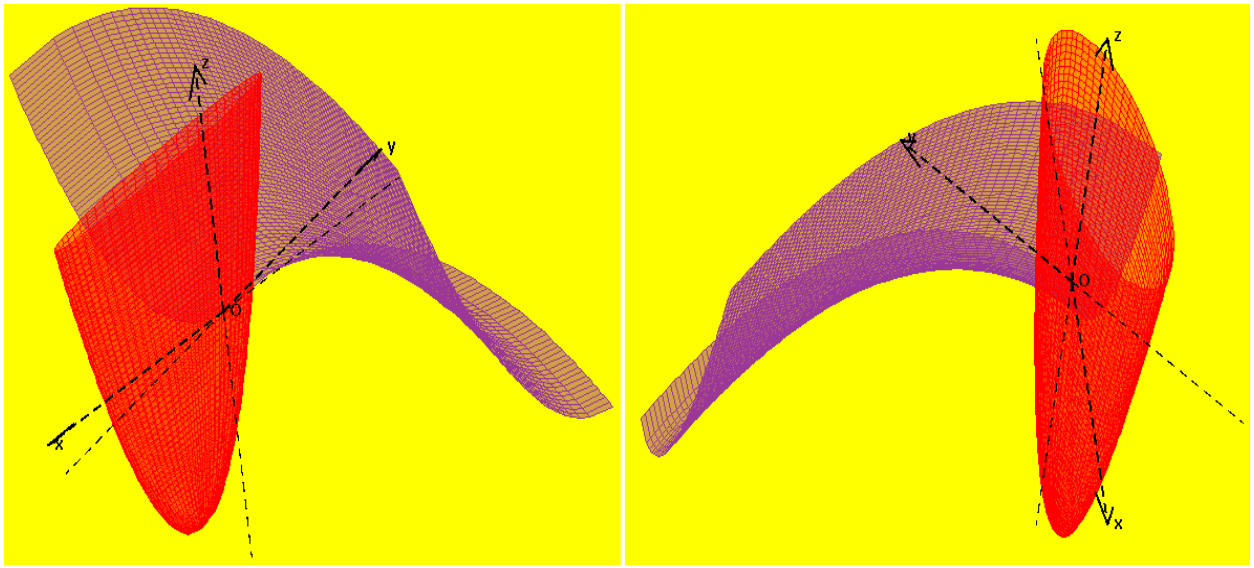


Figura 5.3.3. Desenarea paraboloidului eliptic şi paraboloidului hiperbolic

5.3.3 Modalităţi de determinare a unui paraboloid

Ecuaţia generală ce defineşte un paraboloid eliptic cu vârful în punctul de coordonate (x_0, y_0, z_0) şi de parametrii *a*, *b* şi *p* este [110]:

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=2\cdot p\cdot z. \tag{5.3.5}$$

Ecuaţia generală ce defineşte un paraboloid hiperbolic cu vârful în punctul de coordonate (x_0, y_0, z_0) şi de parametrii *a*, *b* şi *p* este [110]:

$$\frac{(x-x_0)^2}{a^2}-\frac{(y-y_0)^2}{b^2}=2\cdot p\cdot z. \tag{5.3.6}$$

O variantă de a defini un paraboloid este prin specificarea celor trei parametrii şi a coordonatelor vârfului. În figura 5.3.3 sunt reprezentate mai multe astfel de cuadrice prin utilizarea următoarelor două proiecţii: $(\alpha=-145, \beta=45, \gamma=100)$ şi $(\alpha=-80, \beta=145, \gamma=80)$.

5.3.4 Planul tangent şi normala într-un punct la elipsoid

Ecuaţia planului tangent într-un punct $M(x_1(u,v), y_1(u,v), z_1(u,v))$ la un elipsoid este [27]:

$$p\cdot x+q\cdot y+r\cdot z-(p\cdot x_1+q\cdot y_1+r\cdot z_1)=0,$$

$$\text{unde}\quad \begin{cases} p=-bc\cdot cos^2 u\cdot cos\, v \\ q=-ac\cdot cos^2 u\cdot cos\, v. \\ r=-ab\cdot sin\, u\cdot cos\, u \end{cases} \tag{5.3.7}$$

Ecuaţia normalei se obţine pornind de la proprietatea că vectorul director al acesteia reprezintă vectorul normal al planului tangent la cuadrică în acelaşi punct. În figura 5.3.4 sunt prezentate planul tangent şi normala la elipsoid într-un punct specificat. Pentru desenarea acestor drepte s-au utilizat metode ale claselor: *Dreapta3D, Plan3D* şi *Elipsoid*.

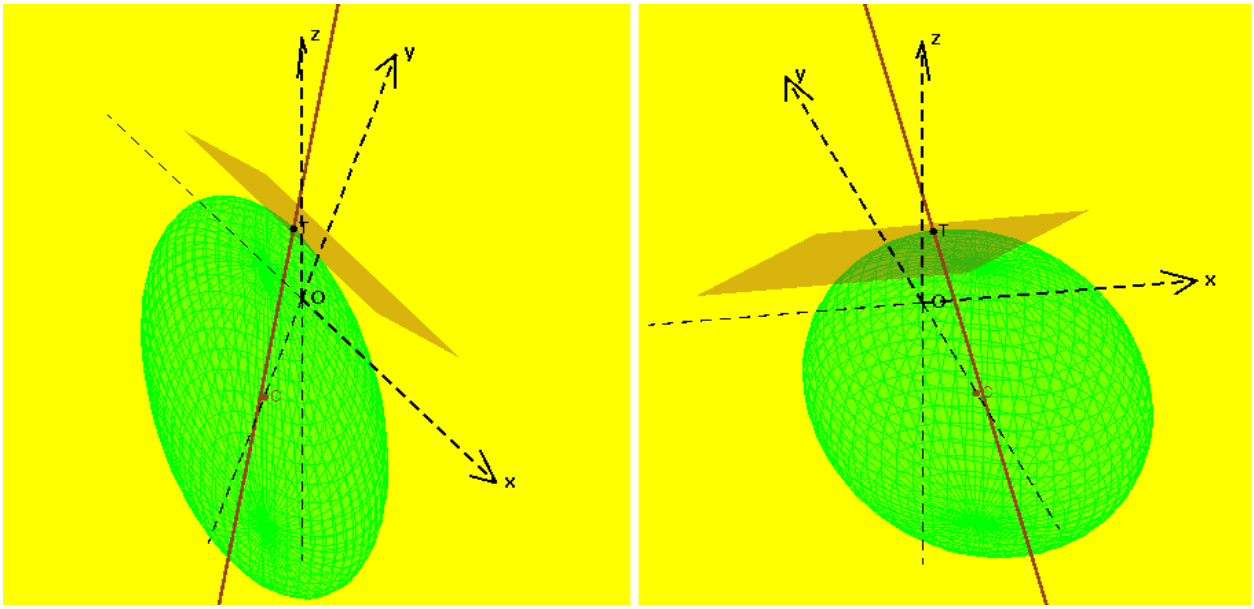


Figura 5.3.4. Planul tangent şi normala într-un punct la elipsoid

În partea de prezentare de noțiuni teoretice se poate observa că există opțiune pentru determinarea primei forme pătratice fundamentale specifice unui elipsoid, dar şi a celei de-a doua forme pătratice fundamentale.

5.3.5 Planul tangent şi normala într-un punct la hiperboloid

Ecuația planului tangent într-un punct $M(x_1(u,v), y_1(u,v), z_1(u,v))$ la un hiperboloid cu o pânză este [27]:

$$p \cdot x + q \cdot y + r \cdot z - (p \cdot x_1 + q \cdot y_1 + r \cdot z_1) = 0,$$

$$\text{unde} \quad \begin{cases} p = -bc \cdot \cosh^2 u \cdot \cos v \\ q = -ac \cdot \cosh^2 u \cdot \sin v \\ r = ab \cdot \sinh u \cdot \cosh u \end{cases}. \tag{5.3.8}$$

Ecuația planului tangent într-un punct $M(x_1(u,v), y_1(u,v), z_1(u,v))$ la un hiperboloid cu două pânze este [73]:

$$p \cdot x + q \cdot y + r \cdot z - (p \cdot x_1 + q \cdot y_1 + r \cdot z_1) = 0,$$

$$\text{unde} \quad \begin{cases} p = -bc \cdot \sinh^2 u \cdot \cos v \\ q = -ac \cdot \sinh^2 u \cdot \sin v \\ r = ab \cdot \sinh u \cdot \cosh u \end{cases}. \tag{5.3.9}$$

Ecuația normalei se obține pornind de la proprietatea că vectorul director al acesteia reprezintă vectorul normal al planului tangent la cuadrică în acelaşi punct.

5.3.6 Planul tangent şi normala într-un punct la paraboloid

Ecuația planului tangent într-un punct $M(x_1(u,v), y_1(u,v), z_1(u,v))$ la un paraboloid eliptic este [27]:

$$p \cdot x + q \cdot y + r \cdot z - (p \cdot x_1 + q \cdot y_1 + r \cdot z_1) = 0,$$

$$\text{unde} \quad \begin{cases} p = b \cdot \sqrt{2pu} \cdot cos\, v \\ q = -a \cdot \sqrt{2pu} \cdot sin\, v. \\ r = abp \end{cases} \tag{5.3.10}$$

Ecuația planului tangent într-un punct $M(x_1(u,v), y_1(u,v), z_1(u,v))$ la un paraboloid hiperbolic este [73]:

$$p \cdot x + q \cdot y + r \cdot z - (p \cdot x_1 + q \cdot y_1 + r \cdot z_1) = 0,$$

$$\text{unde} \quad \begin{cases} p = -b \cdot \sqrt{\dfrac{p}{2v}} \\ q = a \cdot \sqrt{\dfrac{p}{2u}} \\ r = \dfrac{abp}{2\sqrt{uv}} \end{cases}. \tag{5.3.11}$$

Ecuația normalei se obține pornind de la proprietatea că vectorul director al acesteia reprezintă vectorul normal al planului tangent la cuadrică în același punct. În figura 5.3.5 sunt prezentate planul tangent şi normala la un paraboloid eliptic într-un punct specificat prin utilizarea următoarelor două proiecții: ($\alpha = -115, \beta = -15, \gamma = 90$) şi ($\alpha = 80, \beta = 170, \gamma = 90$).

Pentru desenarea acestor drepte s-au utilizat metode ale claselor: *Dreapta3D, Plan3D* şi *Paraboloid*. În partea de prezentare de noțiuni teoretice se poate observa că există opțiune pentru determinarea primei forme pătratice fundamentale specifice unui elipsoid, dar şi a celei de-a doua forme pătratice fundamentale.

5.3.7 Proiecții ortogonale

Sistemul interactiv permite utilizarea oricărei proiecții ortogonale pentru reprezentarea elementelor geometrice 3D ca imagini bidimensionale în planul de proiecție. Specificarea proiecției dorite se face prin intermediul celor trei unghiuri pe care le fac cele trei axe ale reperului ortonormat din spațiu cu axa Ox din planul de proiecție.

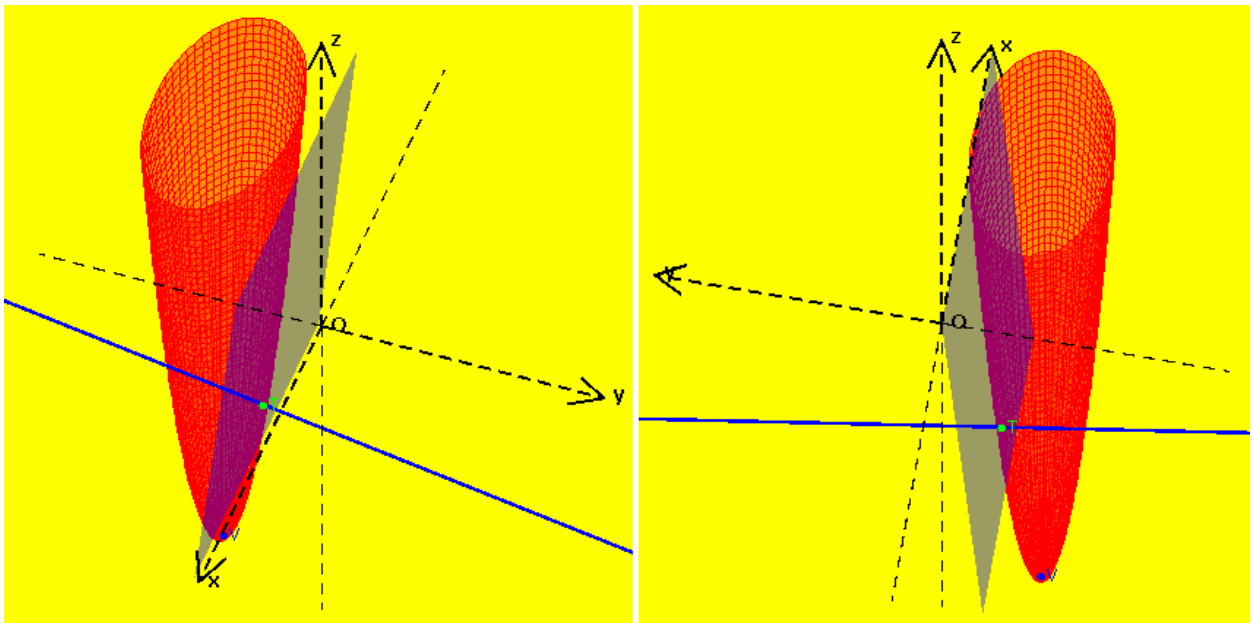


Figura 5.3.5. Planul tangent şi normala într-un punct la paraboloidul eliptic

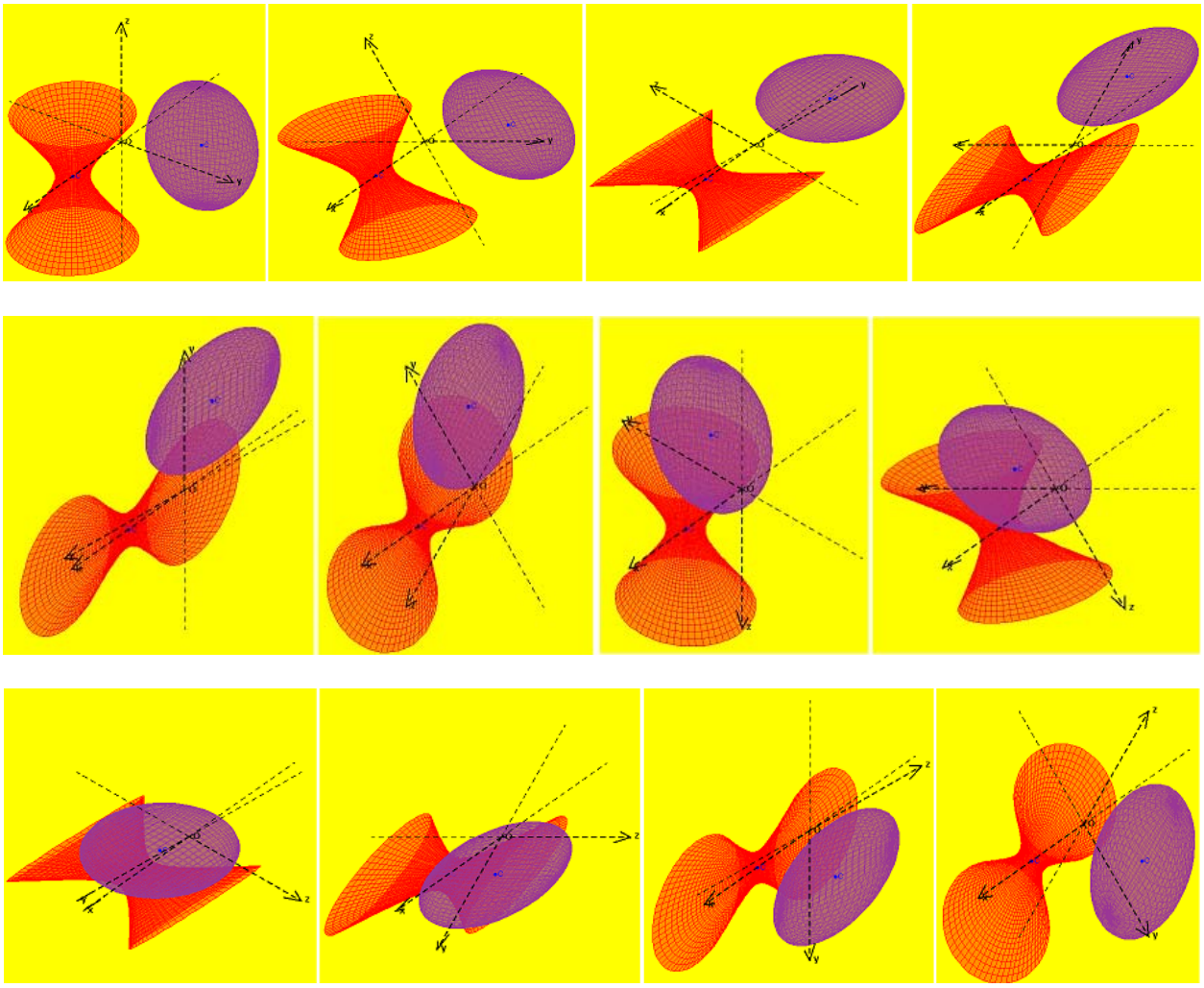


Figura 5.3.6. Animație prin rotație în jurul axei Ox

După realizarea unei construcții geometrice în spațiu există posibilitatea schimbării punctului de observare prin rotații în jurul axelor Ox, Oy şi Oz.

În figura 5.3.6 este prezentată o secvență din animația unui elipsoid şi a unui hiperboloid cu o pânză, animație realizată prin rotație în jurul axei Ox. Secvența prezentată cuprinde 12 imagini bidimensionale obținute prin utilizarea următoarelor proiecții: $(\alpha=-145,\beta=-30,\gamma=90)$, $(\alpha=-145,\beta=0,\gamma=120)$, $(\alpha=-145,\beta=30,\gamma=150)$, $(\alpha=-145,\beta=60,\gamma=180)$, $(\alpha=-145,\beta=90,\gamma=-150)$, $(\alpha=-145,\beta=120,\gamma=-120)$, $(\alpha=-145,\beta=150,\gamma=-90)$, $(\alpha=-145,\beta=180,\gamma=-60)$, $(\alpha=-145,\beta=-150,\gamma=-30)$, $(\alpha=-145,\beta=-120,\gamma=0)$, $(\alpha=-145,\beta=-90,\gamma=30)$ şi $(\alpha=-145,\beta=-60,\gamma=60)$.

5.4 Transformări geometrice

Sistemul interactiv permite aplicarea unor izometrii particulare elementelor geometrice din planul şi spațiul euclidian. Aceste izometrii sunt: translația, rotația şi simetria.

În figura 5.4.1 sunt prezentate două patrulatere: un dreptunghi şi un romb obținute prin rotația de unghi 120^o în raport cu punctul C. În figura 5.4.2 sunt prezentate elipsa obținută prin rotația de unghi 90^o a unei elipse date şi hiperbola obținută prin rotația de unghi 60^o a unei hiperbole date.

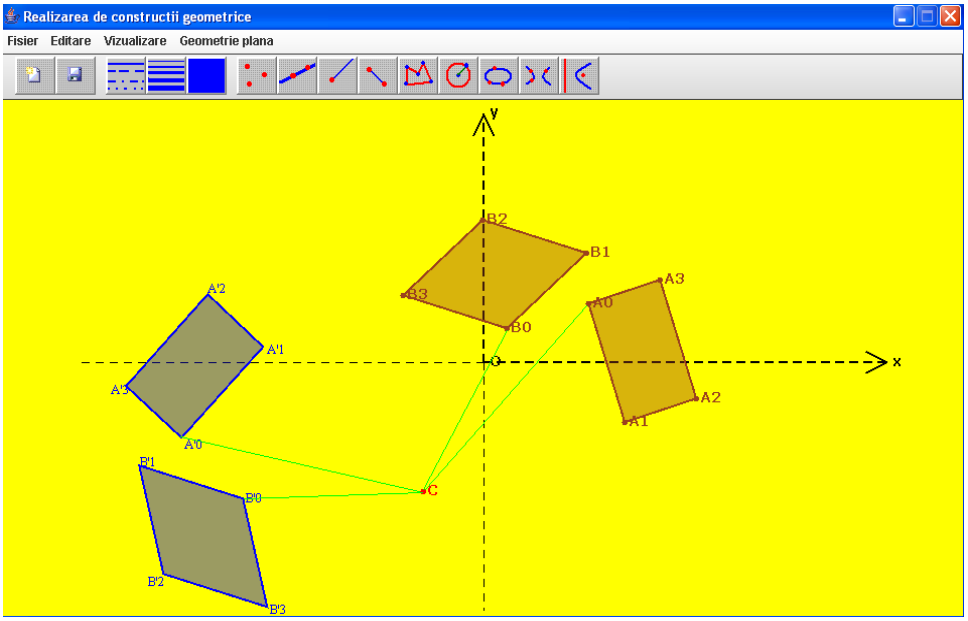


Figura 5.4.1. Rotația unui dreptunghi şi a unui romb

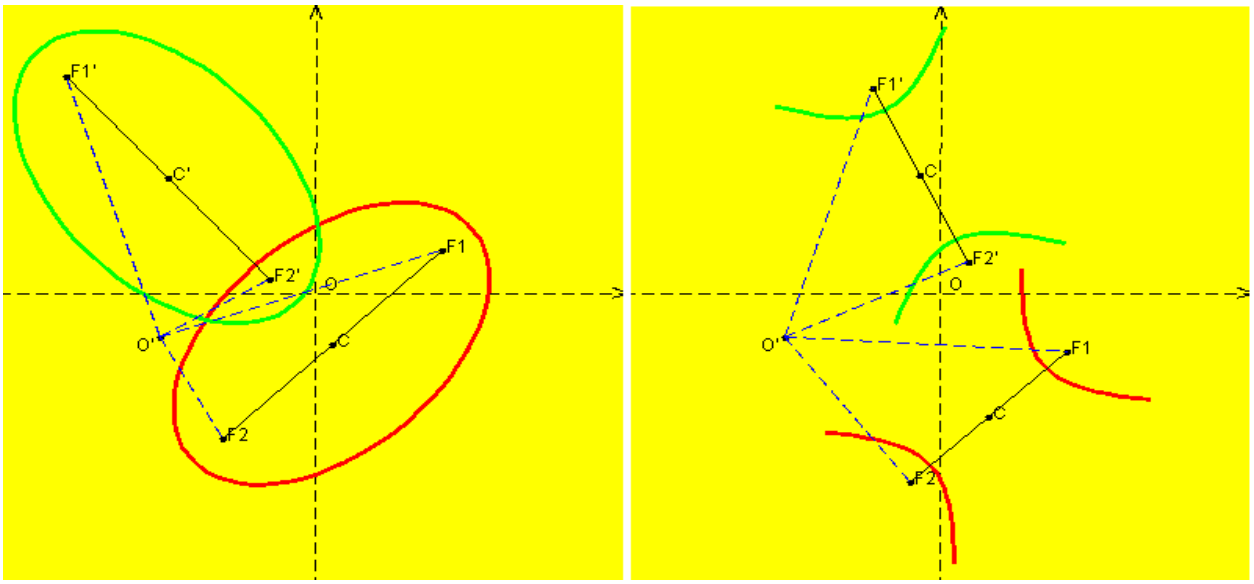


Figura 5.4.2. Rotația unei elipse şi a unei hiperbole

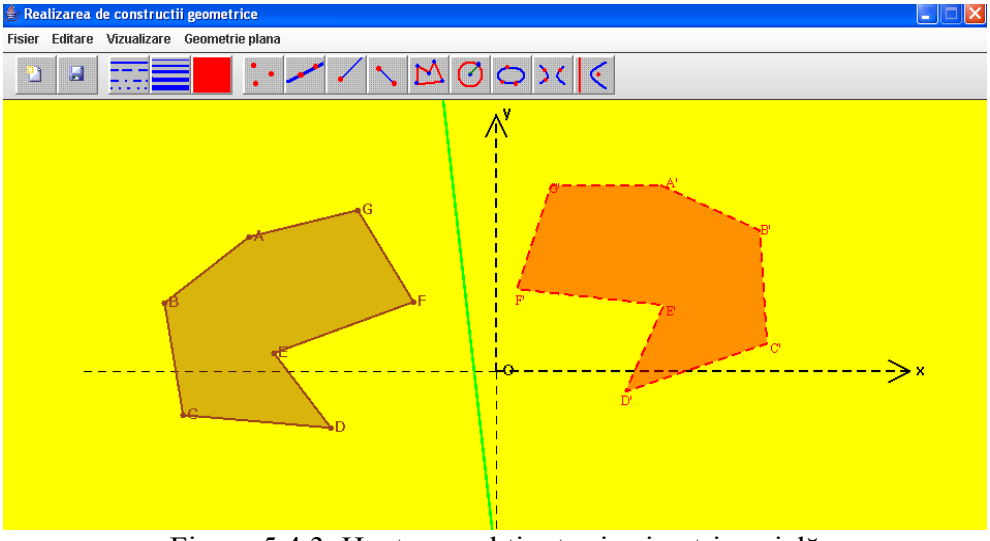


Figura 5.4.3. Heptagon obținut prin simetrie axială

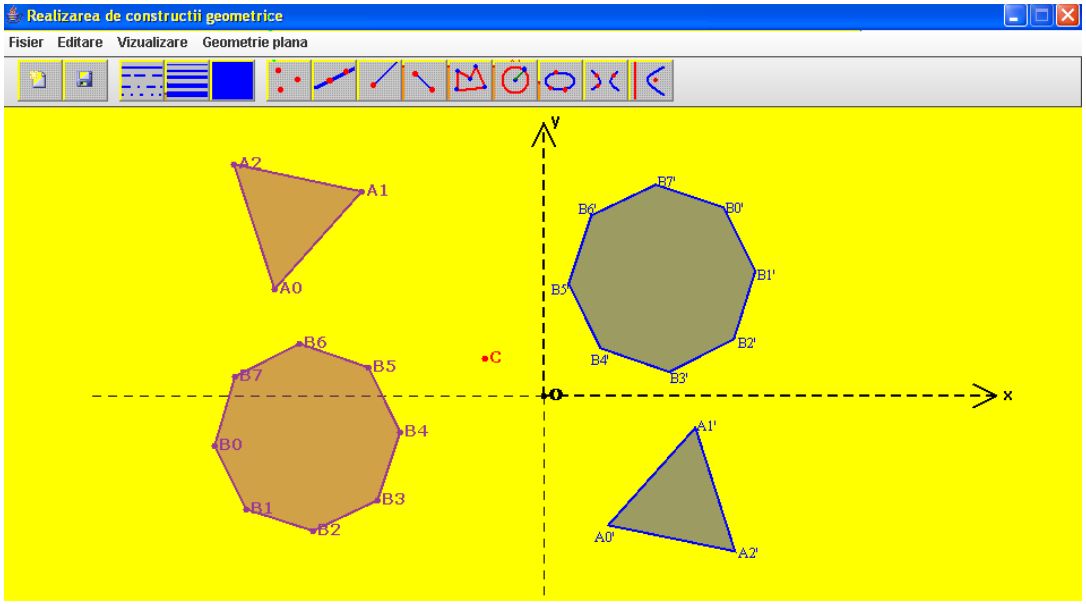


Figura 5.4.4. Triunghi echilateral şi octogon regulat obținut prin simetrie centrală

În figura 5.4.3 este prezentat un heptagon obținut prin simetrie axială, iar în figura 5.4.4 sunt reprezentate două poligoane regulate: triunghi echilateral şi octogon obținute prin simetrie centrală.

În figura 5.4.5 este prezentat elipsoidul obținut prin rotația în planul x0y de unghi -120^{o} a unui elipsoid construit inițial, dar şi elipsoidul obținut prin rotația în planul y0z de unghi 90^{o} a aceluiaşi elipsoid construit inițial. La realizarea imaginilor bidimensionale ale construcției geometrice în spațiu s-au utilizat următoarele două proiecții: ($\alpha=-120,\beta=-150,\gamma=90$) şi ($\alpha=-85,\beta=30,\gamma=90$).

Aplicația interactivă permite rotații în planele xOy, xOz şi yOz ale elementelor geometrice din spațiu, dar şi în plane paralele cu unul dintre cele trei specificate anterior. Ecuația matricială utilizată pentru rotația de unghi θ în planul xOy în raport cu centrul $C(x_C,y_C,z_C)$ este [104]:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} + \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x - x_C \\ y - y_C \\ z - z_C \end{pmatrix}. \qquad (5.4.1)$$

Ecuația matricială utilizată pentru rotația de unghi θ în planul xOz în raport cu centrul $C(x_C,y_C,z_C)$ este [104]:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} + \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x - x_C \\ y - y_C \\ z - z_C \end{pmatrix}. \qquad (5.4.2)$$

Ecuația matricială utilizată pentru rotația de unghi θ în planul yOz în raport cu centrul $C(x_C,y_C,z_C)$ este [104]:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x - x_C \\ y - y_C \\ z - z_C \end{pmatrix}. \qquad (5.4.3)$$

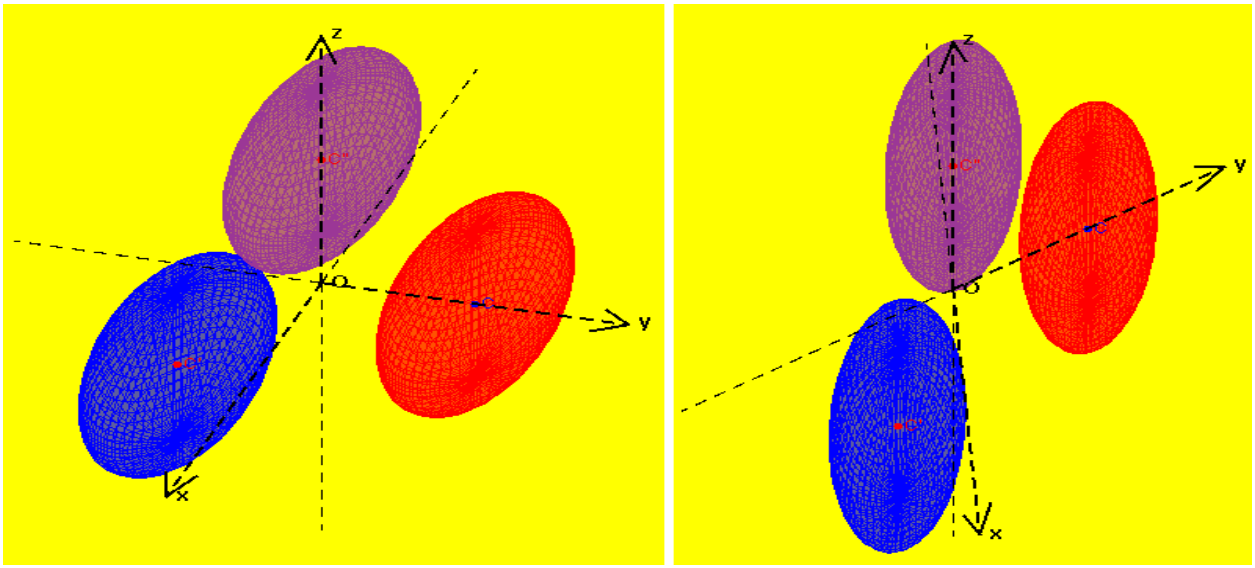


Figura 5.4.5. Rotația unui elipsoid în planele xOy şi yOz

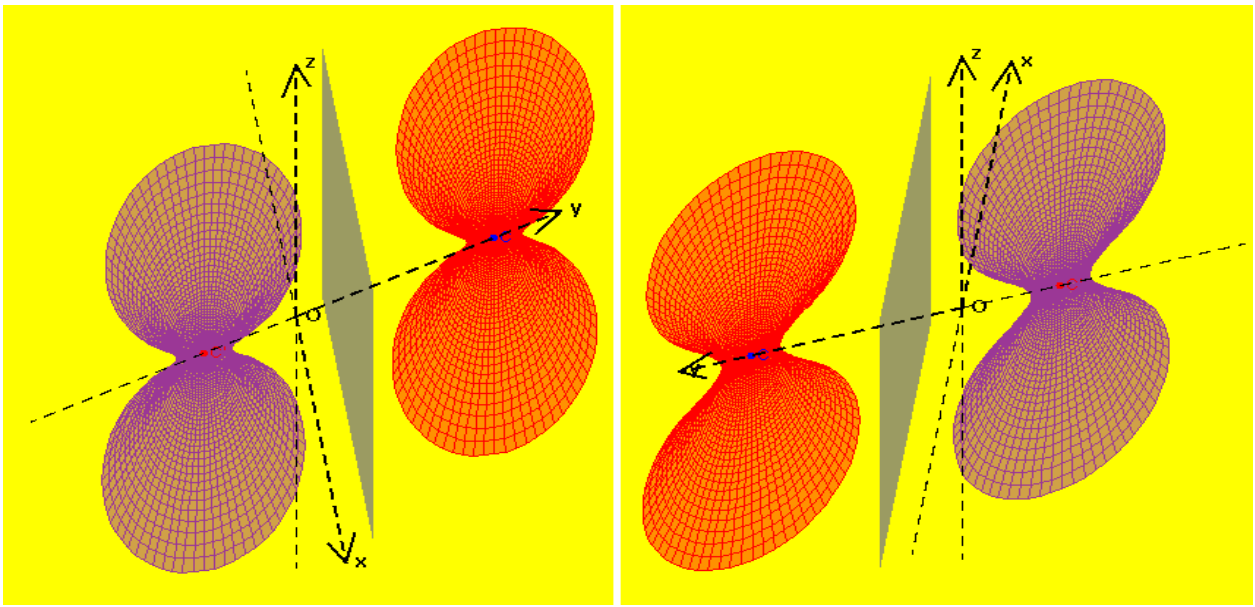


Figura 5.4.6. Simetricul unui hiperboloid cu o pânză față de un plan

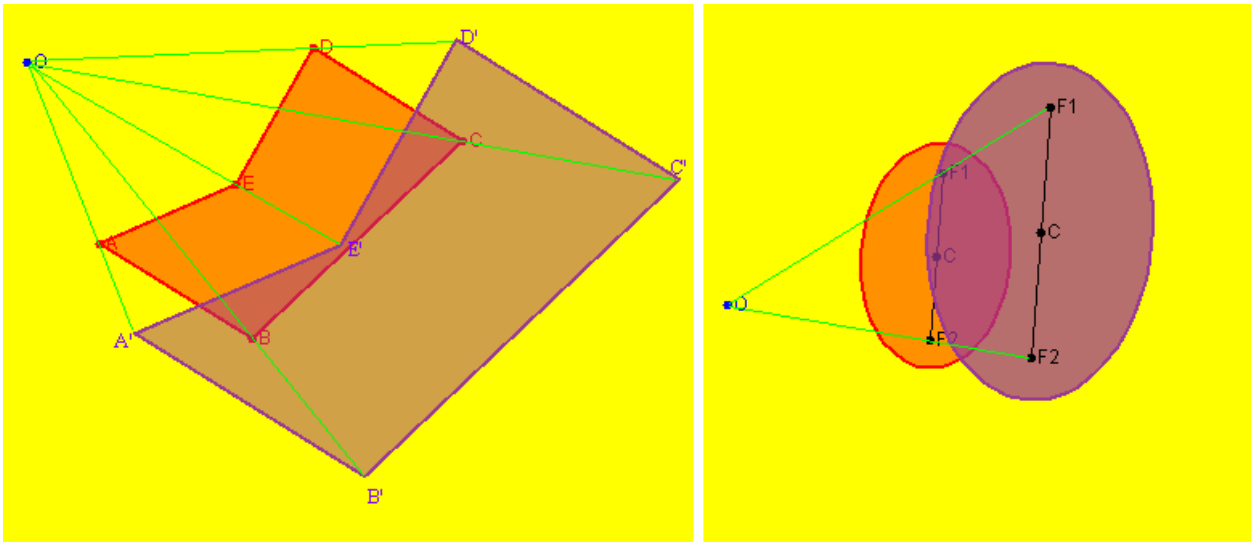


Figura 5.4.7. Omotetie aplicată unui pentagon şi unei elipse

În figura 5.4.6 este prezentat hiperboloidul cu o pânză obținut prin simetria față de un plan paralel cu planul x0y a unui hiperboloidul cu o pânză construit inițial. La realizarea imaginilor bidimensionale ale construcției geometrice în spațiu s-au utilizat următoarele două proiecții: ($\alpha = -80, \beta = 25, \gamma = 90$) şi ($\alpha = 80, \beta = -165, \gamma = 90$).

Sistemul interactiv cuprinde opțiuni şi pentru alte transformări geometrice în plan ce pot fi aplicate elementelor geometrice: omotetia şi inversiunea. În figura 5.4.7 este prezentată omotetia de centru O şi de raport 1.5 aplicată unui pentagon şi unei elipse.

5.5 Concluzii

În acest capitol a fost definită mulțimea operațiilor de bază specifice sistemului dinamic prezentat destinat procesului de asimilare a cunoştințelor din domeniul geometriei.

Comparând funcționalitățile sistemului interactiv prezentat cu două dintre cele mai importante astfel de sisteme existente: Cabri şi Cinderella se pot observa contribuțiile aduse prin implementarea unor opțiuni inexistente la aceste două sisteme dinamice.

În cazul geometriei plane, sistemul realizat cuprinde două secțiuni:

- prezentarea noțiunilor şi rezultatelor principale corespunzătoare noțiunilor: triunghi, patrulater şi vectori;
- realizarea de construcții geometrice exacte.

Prima secțiune, formată din trei interfețe grafice prezentate în figurile 4.3.4, 4.3.5 şi 4.3.6 din capitolul anterior, permite prezentarea rezultatelor teoretice însoțite de construcțiile geometrice aferente. Aceste desene sunt realizate dinamic, astfel că prin selectarea unui punct liber şi modificarea coordonatelor acelui vârf (prin deplasare) se redesenează automat întreaga figură. Această secțiune reprezintă una din contribuțiile personale, ea nefiind prezentă în cazul celor două sisteme informatice amintite anterior.

În cazul celei de-a doua secțiuni, acest sistem cuprinde câteva opțiuni inexistente în cazul sistemelor Cabri şi Cinderella, cum ar fi:

- Centrul de greutate, ortocentrul, medianele şi înălțimile unui triunghi (figura 5.1.5);
- Punctul lui Lemoine şi simedianele unui triunghi (figura 5.1.6);
- Cercul înscris şi cercurile exînscrise unui triunghi (figura 5.1.11);
- Cercul lui Euler (figura 5.1.12) şi primul cerc al lui Lemoine (figura 5.1.13);
- Tangenta şi normala într-un punct la elipsă (figura 5.2.5);
- Tangenta şi normala într-un punct la parabolă (figura 5.2.8);
- Omotetia aplicată tuturor elementelor geometrice 2D (figura 5.4.5).

În cazul geometriei analitice în spațiu, sistemul realizat cuprinde două secțiuni:

- prezentarea noțiunilor şi rezultatelor principale corespunzătoare cuadricelor;
- realizarea de construcții geometrice exacte.

Această parte reprezintă o altă noutate adusă de acest sistem interactiv, ea nefiind prezentă în cazul celor două sisteme informatice amintite anterior.

Prima secțiune reprezentată de interfața grafică prezentată în figura 4.3.7 din capitolul anterior, permite prezentarea rezultatelor teoretice însoțite de construcțiile geometrice aferente. Există opțiuni care permit determinarea primei şi celei de-a doua forme pătratice fundamentale specifice tipului de cuadrică reprezentat, precum şi desenarea planului tangent şi a normalei când se specifică un punct al cuadricii.

În cazul celei de-a doua secțiuni, acest sistem cuprinde câteva opțiuni cum ar fi:

- determinarea punctului dat de intersecția dintre o dreaptă şi un plan;

- determinarea dreptei determinată de intersecția a două plane;
- desenarea cuadricelor: elipsoid (figura 5.3.1), hiperboloid (figura 5.3.2), paraboloid (figura 5.3.3), sfera;
- desenarea pseudosferei şi a elicoidului;
- reprezentarea planul tangent şi a normalei la elipsoid într-un punct (figura 5.3.4);
- reprezentarea planul tangent şi a normalei la hiperboloid într-un punct;
- reprezentarea planul tangent şi a normalei la paraboloid într-un punct (figura 5.3.5);
- reprezentarea planul tangent şi a normalei la sferă într-un punct;
- aplicarea transformărilor geometrice: translație, rotație şi simetrie tuturor elementelor 3D (figurile 3.4.3 şi 3.4.4);
- animații în jurul celor trei axe (figura 5.3.6);
- vizualizarea elementelor geometrice 3D ca imagini bidimensionale în planul de proiecție prin utilizarea proiecției ortogonale dorite.

CONCLUZII, CONTRIBUȚII PERSONALE ŞI PERSPECTIVE

6.1 Concluzii

Obiectivul principal al tezei de doctorat este proiectarea şi implementarea unui mediu dinamic interactiv destinat studiului geometriei. În cadrul cercetărilor pentru conceperea şi construirea sistemului interactiv a fost respectată programa şcolară, dar încorporează şi învățământul la distanță.

Avantajele sistemului informatic interactiv realizat sunt:

- Prin multitudinea de facilități oferite, acest soft matematic va putea fi utilizat cu succes pentru instruirea asistată de calculator la geometrie atât în mediu preuniversitar, cât şi universitar;
- Va fi util atât în etapa de însuşire de noi cunoştințe cât şi în etapa de consolidare a cunoştințelor dobândite;
- S-a realizat luându-se în considerație toate cerințele metodice;
- Informațiile vor fi prezentate într-un mod interesant şi atractiv, accesul la informații realizându-se în mod gradual;
- Prin multitudinea de opțiuni oferite, aplicația va înlocui cu succes hârtia şi creionul pentru realizarea construcțiilor geometrice precise în plan şi în spațiu;
- Programul este preconizat pentru utilizarea lui în procesul de instruire, dar este posibilă şi o utilizare independentă;
- Conținutul ştiințific prezentat va fi bine structurat, existând material suplimentar.

Utilizarea sistemului informatic interactiv în studierea geometriei va contribui la formarea şi dezvoltarea culturii informaționale ale elevilor. Instruirea asistată de calculator în procesul de studiu al elementelor de geometrie este şi o metodă eficientă de sporire a motivației învățării acestei discipline şi a calității însuşirii ei.

Tema tratată în cadrul aplicației tezei este de mare actualitate, geometria fiind o componentă importantă în formarea tinerilor matematicieni, ingineri sau arhitecți. Aplicația informatică a tezei de doctorat se înscrie în această problematică majoră şi de certă actualitate, pe care o abordează într-o manieră nouă, prin prisma tehnologiilor educaționale moderne. Prin utilizarea metodelor specifice învățământului la distanță, teza de doctorat se constituie într-o abordare multidisciplinară.

6.2 Contribuții personale

Contribuția tezei se referă la aspectele de pedagogie inginerească informatică şi care cuprind analiza şi conceperea următoarelor componente ale instruirii asistate de calculator:

- Interfața utilizator – sistem informatic;
- Structurarea şi organizarea materialului predat (de instruire);
- Navigabilitatea în domeniul de cunoştințe.

Un scurt rezumat al contribuțiilor aduse în lucrarea de față este prezentat în continuare.

- Capitolul 2

În acest capitol au fost prezentate câteva contribuții personale la realizarea unor programe educaționale destinate studierii unor capitole din:

- matematică: distribuția binomială, numere complexe, grafice de funcții, locuri geometrice elementare, distribuția normală, după cum s-a publicat şi în articolele [46, 47, 48, 49, 50];
- fizică: legile gazului ideal, puterea electrică, ciocniri perfect elastice şi plastice, rezultatele obținute fiind publicate în articolele [51, 82];
- informatică: metoda „Divide-et-impera", gramatici stohastice, listele liniare, după cum s-a publicat şi în articolele [45, 52, 86].

- Capitolul 3

În acest capitol s-au reprezentat diagramele corespunzătoare celor trei faze: analiză, proiectare şi implementare, sistemul informatic interactiv fiind astfel descris într-o manieră clară şi concisă. Utilizarea limbajului de modelare UML în realizarea diagramelor este caracterizată de rigoare sintactică, semantică bogată şi suport pentru modelarea vizuală.

Diagramele au fost realizate printr-o abordare într-o manieră nouă, multidisciplinară a aplicației informatice, înglobând atât metodele pedagogiei moderne, cât şi componentele specifice disciplinei de studiat. S-a realizat astfel legătura dintre acțiunile didactice şi scopurile şi obiectivele ştiințific stabilite, prin elaborarea a noi metode şi prin asimilarea a noi mijloace, capabile să sporească randamentul şcolar, permițând elevilor şi studenților să-şi însuşească sistemul cerut de cunoştințe şi tehnici de aplicare a acestora în condiții cât mai optime.

Comparând diagramele prezentate în capitolul 3 cu descrierea atelierului virtual prezentat în [70], se constată că modelarea sistemul interactiv permite:

- Definirea concretă a fiecărui caz de utilizare prin prezentarea diagramelor de activități;
- Implementarea eficientă a claselor corespuzătoare tuturor elementelor geometrice prin realizarea de relații de moştenire, agregare şi compunere;
- Descrierea stărilor prin care trece un obiect şi a tranzițiilor dintre aceste stări;
- Descrierea interacțiunilor dintre obiecte în contexte diferite prin realizarea diagramelor de colaborare, diagrame ce nu sunt prezentate în cazul atelierului virtual anterior amintit;
- Vizualizarea modului în care sistemul este divizat şi a dependențelor dintre module prin realizarea diagramelor de componente, diagrame inexistente în cazul atelierului virtual pentru geometrie amintit anterior.

- Capitolul 4

Acest capitol cuprinde:

- o justificare a limbajului de programare ales;
- prezentarea structurilor de date eficiente utilizate pentru geometria dinamică;
- prezentarea interfeței grafice a sistemului interactiv, precum şi a caracteristicilor ferestrelor corespunzătoare celor două cazuri de realizare de construcții geometrice, în plan şi în spațiu.

Comparându-se interfața grafică corespunzătoare opțiunii de realizare a construcțiilor geometrice în plan prezentată în figurile 4.3.1 şi 4.3.2 cu interfața grafică a sistemului interactiv Cabri [21], prezentat în capitolul al doilea, se poate observa că noua interfață cuprinde câteva intrumente şi opțiuni ale meniului *Geometrie plana* inexistente în cazul sistemului interactiv Cabri, cum ar fi: triunghi, triunghi_echilateral, patrulater, paralelogram, dreptunghi, romb, pătrat, trapez, cerc determinat de trei puncte, elipsă determinată de focare şi un parametru, hiperbolă determinată de focare şi un parametru, normala şi tangenta într-un punct la o conică, mediana şi simediana unui triunghi corespunzătoare unui vârf, precum şi transformări geometrice aplicate tuturor elementelor geometrice elementare în plan. Față de acelaşi sistem Cabri, se poate observa că sistemul propus în această teză permite realizarea de construcții geometrice în spațiu, interfața grafică fiind prezentată în figura 4.3.3.

Comparându-se aceeaşi interfaţă grafică corespunzătoare opţiunii de realizare a construcţiilor geometrice în plan prezentată în figurile 4.3.1 şi 4.3.2 cu interfaţa grafică a sistemului interactiv Cinderella [94], prezentat în capitolul al doilea, se poate observa că noua interfaţă cuprinde câteva intrumente şi opţiuni ale meniului *Geometrie plana* inexistente în cazul sistemului interactiv Cinderella, cum ar fi: centrul cercului circumscris şi centrul cercului înscris într-un triunghi, ortocentrul şi centrul de greutate al unui triunghi, centrul cercului lui Euler, intersecţia a două drepte, mediatoarea unui segment, normala şi tangenta într-un punct la o conică, mediana şi simediana unui triunghi corespunzătoare unui vârf, cercul înscris şi cercul circumscris unui triunghi, cercurile exînscrise pentru un triunghi oarecare, primul cerc al lui Lemoine etc.

Interfaţa sistemului interactiv permite utilizarea acestuia într-un mod simplu şi elegant atât de către elevi sau studenţi, cât şi de cadre didactice. Sistemul informatic prezentat conduce subiectul care îl utilizează la obţinerea unei experienţe în înţelegerea şi stăpânirea de cunoştinţe din domeniul geometriei şi oferă accesul comod şi eficient la informaţiile şi cunoştinţele cele mai noi.

- Capitolul 5

În acest capitol s-au definit mulţimea operaţiilor de bază specifice unui sistem dinamic pentru geometrie şi au fost descrise explicit cum pot fi efectuate în coordonate carteziene calculele necesare operaţiilor de bază.

În cazul geometriei plane, sistemul realizat cuprinde două secţiuni:

- prezentarea noţiunilor şi rezultatelor principale corespunzătoare noţiunilor: triunghi, patrulater şi vectori;
- realizarea de construcţii geometrice exacte.

Prima secţiune, formată din trei interfeţe grafice, permite prezentarea rezultatelor teoretice însoţite de construcţiile geometrice aferente. Aceste desene sunt realizate dinamic, astfel că prin selectarea unui punct liber şi modificarea coordonatelor acelui vârf (prin deplasare) se redesenează automat întreaga figură. Această secţiune reprezintă una din contribuţiile personale, ea nefiind prezentă în cazul celor două sisteme informatice amintite anterior. În cazul celei de-a doua secţiuni, acest sistem cuprinde câteva opţiuni inexistente în cazul sistemelor Cabri şi Cinderella.

În cazul geometriei analitice în spaţiu, sistemul realizat cuprinde două secţiuni:

- prezentarea noţiunilor şi rezultatelor principale corespunzătoare cuadricelor;
- realizarea de construcţii geometrice exacte.

Prima secţiune permite prezentarea rezultatelor teoretice însoţite de construcţiile geometrice aferente. Există opţiuni care permit determinarea primei şi celei de-a doua forme pătratice fundamentale specifice tipului de cuadrică reprezentat, precum şi desenarea planului tangent şi a normalei când se specifică un punct al cuadricii.

Activitatea de cercetare în acest domeniu s-a finalizat prin următoarele contribuţii personale:

- implementerea eficientă a claselor corespunzătoare elementelor geometrice în plan şi în spaţiu prin realizarea de relaţii de moştenire, agregare şi compunere, după cum s-a arătat şi în articolul [57];
- implementarea unor algorimi de intanţiere a elementelor, de intersecţie sau de tangenţă utilizând metode speciale de programare, algoritmi ce contribuie la rapiditatea şi exactitatea reprezentării grafice a elementelor geometrice dorite, algoritmi prezentaţi şi în articolul [52];
- descrierea interacţiunilor dintre obiecte în contexte diferite, precum şi vizualizarea modului în care sistemul este divizat şi a dependenţelor dintre module;

- introducerea de opțiuni inexistente în cazul altor astfel de sisteme pentru realizarea de construcții geometrice exacte în plan, după cum s-a arătat şi în articolele [55,56,58]:
  - centrul de greutate, ortocentrul, medianele şi înălțimile unui triunghi;
  - punctul lui Lemoine şi simedianele unui triunghi;
  - cercul înscris şi cercurile exînscrise unui triunghi;
  - cercul lui Euler şi primul cerc al lui Lemoine;
  - tangenta şi normala într-un punct la elipsă;
  - tangenta şi normala într-un punct la parabolă;
  - omotetia aplicată tuturor elementelor geometrice 2D;
- posibilitatea realizării de construcții geometrice în spațiu, opțiuni prezentate în articolul [54]:
  - determinarea punctului dat de intersecția dintre o dreaptă şi un plan;
  - determinarea dreptei determinată de intersecția a două plane;
  - desenarea cuadricelor: elipsoid, hiperboloid, paraboloid, sfera;
  - desenarea pseudosferei şi a elicoidului;
  - reprezentarea planul tangent şi a normalei la elipsoid într-un punct;
  - reprezentarea planul tangent şi a normalei la hiperboloid într-un punct;
  - reprezentarea planul tangent şi a normalei la paraboloid într-un punct;
  - reprezentarea planul tangent şi a normalei la sferă într-un punct;
  - aplicarea transformărilor geometrice: translație, rotație şi simetrie tuturor elementelor 3D;
- prezentarea noțiunilor şi rezultatelor principale corespunzătoare unor capitole din geometrie însoțite de construcțiile geometrice corespunzătoare;
- vizualizarea elementelor geometrice 3D ca imagini bidimensionale în planul de proiecție prin utilizarea proiecției ortogonale dorite.

6.3 Perspective

În ceea ce privește direcțiile viitoare de cercetare, se va urmări realizarea unui sistem expert capabil să rezolve probleme de geometrie euclidiană plană ce va cuprinde următoarele componente:

- componenta de realizare a inferențelor;
- componenta de reprezentare a cunoştințelor geometrice;
- componenta de realizare automată a construcțiilor geometrice;
- interfața inteligentă ce va avea drept scop traducea problemei din limbaj matematic în limbajul de lucru al tehnologiei informaționale inteligente şi invers.

Pentru realizarea primei componente se va urmări optimizarea tehnicilor euristice existente, raționamentul fiind unul ghidat de scop deoarece se pleacă de la nişte premise şi se dorește atingerea unui scop bine precizat. Reprezentarea spațiului soluției se va face prin intermediul unui graf ŞI/SAU, aspectul specific care trebuie considerat este cum poate fi utilizată informația euristică în căutarea soluției.

Sistemele informatice de instruire consideră că procesul de predare/învățare este o activitate care are loc în interiorul celui care studiază. În acest sens în viitor este necesar să fie studiate şi rezolvate câteva elemente esențiale ale proiectării acestor sisteme de instruire:

- principii şi metode funcționale pentru structura predării şi prezentării cursurilor;
- corectarea erorilor făcute de student şi asigurarea feedback-ul asupra studentului;
- creearea unui sistem de apreciere şi notare;
- dezvoltarea unui sistem de auto-control şi auto-apreciere pentru a creşte motivația studentului;

- metode de evaluare eficientă a materialelor de predare metode de evaluare a asimilării cunoştinţelor de către student;
- creearea şi coordonarea legăturilor între profesor şi student prin intermediul unui sistem inteligent de instruire;
- metode de menţinere a interesului pentru învăţare de către student;
- metode de a stimula entuziasmul studentului pentru procesul de asimilare a cunoştinţelor.

BIBLIOGRAFIE

[1]. Adăscăliței A., *Instruire asistată de calculator. Didactica informatica*, Editura Polirom, 2007

[2]. Ael Educațional, *Lecții de matematică*, http://www.advancedelearning.com/index.php/articles/c321

[3]. Ahmad A., Salim S.S., Zainuddin R., *A Cognitive Tool to Support Mathematical Communication in Fraction Word Problem Solving***,** WSEAS Transactions on Computers, vol. 7, 2008, pag. 228-236

[4]. Ainley M., *Interest in learning and classroom interaction*, Perspectives on practice and meaning in mathematics and science classroom, D.J. Clarke Ed., Olanda, 2001

[5]. Andrica D., Duca D., Purdea I., Pop I., *Matematica de bază*, Editura Studium, Cluj-Napoca, 2004

[6]. Andrica D., Varga C., Văcărețu D., *Teme şi probleme alese de geometrie*, Editura Plus, București, 2002

[7]. Andrica D., Țopan L., Analytic Geometry, Cluj University Press, 2004

[8]. Arzarello F., Micheletti C., Olivero F., Robutti O., Paola D., Gallino G., *Drawing in Cabri and modalities of transition from conjectures to proof in geometry*, Proceedings of the 22nd Psychology of Mathematics Education Conference, vol. 2, pag. 32-39, 1998

[9]. Bennet S., McRobb S., Farmer R., *Object Oriented Systems Analysis and Design using UML*, McGraw Hill, 1999

[10]. Bloch J., *JAVA, ghid practic pentru programatori avansați*, Editura Teora, Bucureşti, 2002

[11]. Bocu D., *Inițiere în modelarea obiect orientată a sistemelor soft utilizând UML*, Editura Albastră, Cluj-Napoca, 2002

[12]. Bocu D., *Inițiere în ingineria sistemelor soft*, Editura Albastră, Cluj-Napoca, 2002

[13]. Bocu D., Bocu R., *Modelare obiect orientată cu UML*, Editura Albastră, Cluj-Napoca, 2006

[14]. Bontaş I., *Pedagogie. Tratat*, Editura All, 2008

[15]. Booch G., Rumbaugh J., Jacobson I., *The Unified Modeling Language User Guide*, Addison Wesley, 1999

[16]. Brannan D.A., Esplen M. F., Gray J., *Geometry*, Cambridge: Cambridge University Press, 1999

[17]. Brânzei D., Brânzei R., *Metodica predării matematicii*, Editura Paralela 45, 2003

[18]. Brut M., *Instrumente pentru e-learning. Ghidul informatic al profesorului modern*, Editura Polirom, 2006

[19]. Bucoş M., Jucan D., *Teoria şi metodologia instruirii. Teoria şi metodologia evaluării. Repere şi instrumente didactice pentru formarea profesorilor*, Editura Paralela 45, 2007

[20]. Bumbaru S., *Programare orientată pe obiecte în limbajul Java*, Editura Fundației Universitare "Dunărea de Jos", Galați, 2002

[21]. Cabrilog, *Cabri Geometre II Plus*, http://www.cabri.com/v2/pages/en/index.php

[22]. Cerghit I., *Sisteme de instruire alternative şi complementare. Structuri, stiluri şi strategii*, Editura Polirom, 2008

[23]. Company P., Contero M., Piquer A., Aleixos N., Conesa J., Naya F., *Educational software for teaching drawing-based conceptual design skills***,** Computer Applications in Engineering Education, Volumul 12, 2004, pag. 257-268

[24]. Cosmovici A., Iacob L., *Psihologie şcolară*, Editura Polirom, Bucureşti, 1998

[25]. Cucoş C., *Pedagogie*, Editura Polirom, 2002

[26]. Cucoş C., *Informatizarea în educație. Aspecte ale virtualizării formării*, Editura Polirom, 2006

[27]. Dumitras D. E., *Capitole de geometrie analitică şi diferențială*, 2005, Editura Digital Data, Cluj-Napoca

[28]. Eshuis R., Wieringa R., *A formal semantics for UML activity diagrams – Formalising workflow models*, University of Twente, Departament of Computer Sci*ence*, 2001

[29]. Fendt W., *Java Applets on Mathematics*, www.walter-fendt.de/m14e/

[30]. Fortenberry N., Powlik J., *A national digital library for science, mathematics, engineering, and technology education***,** Computer Applications in Engineering Education, vol. 7, 1999, pag. 45-49

[31]. Fowler M., Scott K., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Addison Wesley, Readings MA, USA, 2000

[32]. GeoCentral, *Interactive Geometry*, http://geocentral.net/geometria/

[33]. Glusac D., Radosav D., Karuovic D., Ivin D., *Pedagogical and Didactic-Methodical Aspects of E-learning*, 6th WSEAS International Conference on E-ACTIVITIES, Tenerife, Spania, 2007, pag. 67-75

[34]. Grand M., *A Catalog of Reusable Design Patterns Illustrated with UML*, John Wiley & Sons, 2002

[35]. Guimarães L.C., Barbastefano R.G., Belfort E., *Tools for teaching mathematics: A case for Java and VRML***,** Computer Applications in Engineering Education, Volumul 8, 2000, pag. 157-161

[36]. Hartshorne R., *Companion to Euclid: A course of geometry, based on Euclid's Elements and its modern descendants*, American Mathematical Society, 1997

[37]. Henle M., *Modern Geometries: The Analytic Approach*, Upper Saddle River: Prentice-Hall, 1997

[38]. Hidaka K., *User-interface of a tool for learning geometry*, Computers & Education, Volumul 24, 1995, pag. 59-66

[39]. Hohenwarter M., *GeoGebra*, http://www.geogebra.org/cms/

[40]. Hoyles C., *Microworlds/schoolworlds: the transformation of an innovation*, Learning from computers: mathematics education and technologies, Keitel and Ruthven, Berlin, 1993

[41]. Hölzl R., *Im Zugmodus der Cabri Geometrie*, Universität Augsburg, 1994

[42]. Husin H., Hamid S., Nizam M., *Review on UML CASE Tools***,** Proceedings of the 3rd WSEAS/IASME International Conference on Educational Technologies, Arcachon, Franța, 2007, pag. 347-351

[43]. Iglesias C.E., Carbajo A.G., Sastre Rosa M.A., *Interactive tools for Discrete Mathematics e-learning*, WSEAS Transactions on Advances in Engineering Education, vol. 5, 2008, pag. 97-103

[44]. International Education Software*, Manipula Math Applet Collections*, www.ies.co.jp/math/products

[45]. Iordan A.E., *Pachet de programe software pentru prezentarea metodei « Divide et impera »*, Analele Facultății de Inginerie Hunedoara, 2003, Tomul V, Fascicola 1, pag. 151-158

[46]. Iordan A.E., *Soft educațional pentru prezentarea distribuției binomiale și a aplicațiilor acesteia la definirea texturilor stohastice***,** A X-a Sesiune de Comunicări Științifice cu participare internațională: Leadership și management la orizonturile secolului al XXI-lea*,* Sibiu, Romania, 24-26 Noiembrie 2005, Volumul XII, pag. 192-199, ISBN 973-7809-29-7

[47]. Iordan A., *Pachet de programe software pentru prezentarea numerelor complexe și a aplicațiilor acestora în geometrie*, A X-a Sesiune de Comunicări Științifice cu participare internațională: Leadership și management la orizonturile secolului al XXI-lea*,* Sibiu, Romania, 24-26 Noiembrie 2005, Volumul XII, pag. 200-207, ISBN 973-7809-29-7

[48]. Iordan A., Pănoiu M.**,** *Multimedia educational software for producing graphs of mathematical functions***,** International Journal of Computers, Communications & Control, 2006, Oradea, Romania, Volumul I, pag. 284-289, ISSN 1841-9836 (ISI Reviste)

[49]. Iordan A., Pănoiu M., *Educational software for the presentation of geometrical elementary locus*, XI-*th* International Conference „Man in the Knowledge Based Organization", 23-25 noiembrie 2006, Sibiu, Romania, ISBN 973-7809-51-3, pag. 136-143

[50]. Iordan A., Pănoiu M., *Educational software for exploratory analysis of statistical dates*, XI-*th* International Conference „Man in the Knowledge Based Organization", 23-25 noiembrie 2006, Sibiu, Romania, ISBN 973-7809-51-3, pag. 144-151

[51]. Iordan A., Pănoiu M., *Educational software for presentation of the ideal gas laws,* Annals of the Faculty of Engineering Hunedoara, Romania, 2006, Tomul IV, Fascicola 1, ISSN 1584-2665, pag. 87-92 (revista CNCSIS categoria C)

[52]. Iordan A., Pănoiu M., *The application of stochastic grammars for generation of fractals,* Annals of the Faculty of Engineering Hunedoara, Romania, 2006, Tomul IV, Fascicola 1, ISSN 1584-2665, pag. 133-140 (revista CNCSIS categoria C)

[53]. Iordan A., Pănoiu M., *Dynamical software for the study of Triangle's Geometry*, microCAD 2007 International Scientific Conference, 22-23 martie 2007, Miskolc, Ungaria, pag. 35-40

[54]. Iordan A., Savii G., Pănoiu M., Pănoiu C., *New Software for the Study of the Classic Surfaces from Differential Geometry*, European Computing Conference, Atena, Grecia, 25-27 septembrie 2007, Springer Verlag, pag. 495-504 (ISI Proceedings)

[55]. Iordan A., Pănoiu M., Pănoiu C., *New Dynamical Methods for the Study of the Euclidian Geometry*, International Conference on Education and Educational Technology, Veneția, Italia, 21-23 Noiembrie 2007, pag. 35-40 (ISI Proceedings)

[56]. Iordan A., Savii G., Pănoiu M., Pănoiu C., *Development of a Dynamical Software for Teaching Plane Analytical Geometry*, International Conference on Engineering Education, Heraklion, Grecia, 22-24 iulie 2008, pag. 55-60 (ISI Proceedings)

[57]. Iordan A., Savii G., Pănoiu M., Pănoiu C., *Development of a dynamical software for doing geometrical constructions*, International Conference on Applied Informatics

and Communications, Rodos, Grecia, 20-22 august 2008, pag. 104-109 (ISI Proceedings)

[58]. Iordan A., Savii G., Pănoiu M., Pănoiu C., *Multimedia Interactive Environment for Study the Plane Analytical Geometry*, WSEAS Transactions on Computers, volum 7, octombrie 2008, pag. 1564-1573 (BDI Reviste)

[59]. Iordan A., Savii G., Pănoiu M., Pănoiu C., *Visual interactive environment for doing geometrical constructions*, WSEAS Transactions on Computers, volum 8, februarie 2009, pag. 258-268 (BDI Reviste)

[60]. Iucu R., *Instruirea şcolară. Perspective teoretice şi aplicative*, Editura Polirom, 2008

[61]. Joyce D., *Geometry Applet*, http://aleph0.clarku.edu/~djoyce/java/Geometry/Geometry.html

[62]. Kortenkamp U., Richter-Gebert J., *Cinderella - Erfahrungen mit Java*, Verlag, Heidelberg, 1999

[63]. Kunth D., *Arta programării calculatoarelor – Algoritmi fundamentali*, Editura Teora, Bucureşti, 1999

[64]. Laborde C., *Visual phenomena in the teaching/learning of geometry in a computer-based environment*, Perspectives on the teaching of geometry for the 21st century, 1998, pag. 113-121, Dordrecht, Olanda

[65]. Laborde J. M., *Some issues raised by the development of implemented dynamic geometry as with cabri-géomètre*, Proccedings of the 15th European Workshop on Computational Geometry, 1999

[66]. Lemay L., Cadenhead R., *Java 2 fără profesor în 21 de zile*, Editura Teora, Bucureşti, 2000

[67]. Love E., *The functions of visualization in learning geometry*, Proceedings of NATO Advanced Workshop, 1995, pag. 125-141, Berlin, Springer-Verlag

[68]. Lupu A., Bologa R., Sabău G., Muntean M., Integrated Information Systems in Higher Education, WSEAS Transaction on Computers, vol. 7, 2008, pag. 473-482

[69]. Maciuc I., *Pedagogie I. Repere introductive*, Editura Sitech, 2006

[70]. Macrelle-Rosselle M., *Conception d'un atelier d'expérimentation de logiciels éducatifs. Application on géométrie*, Teză de doctorat, Universitatea „Henri Poincaré", Nancy, Franţa, 2001

[71]. Mark C., Steven W., Griffith A.F., *Java – 1001 secrete pentru programatori*, Editura Teora, Bucureşti, 2002

[72]. Mcdougall A., Squires A., *Empirical study of a new paradigm for choosing educational software*, Computer Education, Elsevier Science, 1995

[73]. Murgulescu E., *Geometrie analitică şi diferenţială*, Editura Didactică şi Pedagogică, Bucureşti, 1965

[74]. Negret-Dobridor I., Panişoara I., *Ştiinţa învăţării. De la teorie la practică*, Editura Polirom, 2005

[75]. Nicola I., *Tratat de pedagogie şcolară*, Editura Aramis, 2003

[76]. Nicolescu L., *Curs de geometrie*, Bucureşti, 2002

[77]. Nicolescu L., Boskoff V., *Probleme practice de geometrie*, Editura Tehnică, Bucureşti, 1990

[78]. Odell J., *Advanced Object Oriented Analysis & Design using UML*, Cambrige University Press, 1998

[79]. Oestereich B., *Developing Software with UML*, Addison Wesley, 1999

[80]. Pavel P., *Environnement distribué interactif pour l'apprentissage humain de la géométrie descriptive*, Teză de doctorat, Universitatea din Maine, 1999

[81]. Panişoara I. O., *Profesorul de succes. 59 de principii de pedagogie practică*, Editura Polirom, 2009

[82]. Pănoiu C., Pănoiu M., Muscalagiu I., Iordan A,, *Visual Interactive Environment for study the power electronics using PSCAD-EMTDC simulation program*, Computer Applications in Engineering Education (acceptată pt. publicare în 1.09.2007) (ISI Reviste)

[83]. Pănoiu M., Iordan A., Pănoiu C., *Multimedia Educational Software for Study Image Formation in Converging and Diverging Lens*, microCAD 2007 International Scientific Conference, 22-23 martie 2007, Miskolc, Ungaria, pag. 41-46

[84]. Pănoiu M., Pănoiu C., Muscalagiu I., Iordan A., *Educational software for a comparative study of direct sort methods*, Annals of the Faculty of Engineering Hunedoara, 2004, Tomul II, Fascicola 2, pag. 177-184

[85]. Pănoiu M., Pănoiu C., Muscalagiu I., Iordan A., *Multimedia educational software for binary tree and their application present*, International Balkan Conference in Informatics, 17-19 noiembrie 2005, Ohrid, Macedonia, pag. 220-227

[86]. Pănoiu M., Pănoiu C., Muscalagiu I., Iordan A., *An interactive learning environment for analyze linked list data structures,* International Journal of Computers, Communications & Control, 2006, Oradea, Romania, Volumul I, pag. 355-359, ISSN 1841-9836 (ISI Reviste)

[87]. Peñín P.A., Morales M.R.P., García D.R., García Díaz R.P., Quirós J.S., *Multimedia-integrated application for computer-assisted teaching of technical drawing***,** Computer Applications in Engineering Education*,* Volumul 12, 2004, pag. 136-144

[88]. Petcu D., *Matematică asistată de calculator*, Editura Eubeea, 2000

[89]. Pintea C., Geometrie, Presa Universitară Clujeană, 2001

[90]. Popescu M., Sterpu M., *Geometrie analitica. Teorie si aplicatii*, 2004, Editura Universitaria, Craiova

[91]. Por L.Y., Zaitun A.B., *An Adaptive User Assessment Model for e-Learning***,** WSEAS Transactions on Advances in Engineering Education, vol. 5, 2008, pag. 158-167

[92]. Prasolov V.V., Tikhomirov V.M., *Geometry*, American Mathematical Society, 2001

[93]. Prenowitz W., Jordan M., *Basic Concepts of Geometry*, New York: Ardsley house Publishers, 1989

[94]. Richter-Gebert J.,Kortenkamp U., *The interactive geometry software Cinderella,* Springer-Verlag, 1999

[95]. Richter-Gebert J.,Kortenkamp U., *Cinderella*, http://cinderella.de/tiki-index

[96]. Roman V., *Geometria 8*, http://www.edusoft.ro/detalii.php?id=27, Editura Edusoft, 2002

[97]. Rosenberg D., Scott K., *Use case Driven Object Modeling with UML*, Addison Wesley, 1999

[98]. Rumbaugh J, Jacobson I., Booch G., *The Unified Modeling Language Reference Manual*, Addison Wesley, 1999

[99]. Ruthven K., Hennessy S., Deaney R., *Constructions of dynamic geometry: A study of the interpretative flexibility of educational software in classroom practice*, Computers & Education, 2007

[100]. Salavastru D., *Psihologia educației*, Editura Polirom, Bucureşti,2004

[101]. Scalon E., Tosunoglu C., Jones A., Butcher P., Ross S., Greenberg J., *Learning with computers : experiences of evaluation*, Computer Education*,* Elsevier Science, 1998

[102]. Siegel A., *PyGeo*, http://pygeo.sourceforge.net/index.html

[103]. Silveira R, Viccari R., *JADE – Java Agents for Distance Education Framework*, 8th Annual International Distance Education Conference, 2001

[104]. Smaranda D., Soare N., *Transformări geometrice*, Bucureşti, 1988

[105]. Softwin, *Geometrie, între joc şi nota 10*, www.intuitext.ro/content.php?page=geometrie

[106]. Trindade J., Fiolhais C., Almeida L., *Science Learning in Virtual Environments*, British Journal of Educational Technology, 2002

[107]. Udrişte C., *Geometrie analitică*, Bucureşti, 1982

[108]. Universitatea din Creta, *Euclidraw*, http://euclidraw.com/

[109]. Vladimirescu I., Popescu M., *Algebra liniara si geometrie analitica*, 1994, Editura Universitaria, Craiova

[110]. Vrănceanu G., Mărgulescu G., *Geometrie analitică*, Editura Didactică şi Pedagogică, Bucureşti, 1973

[111]. Zhao R., Lin L., *An UML Statechart Diagram-Based MM-Path Generation Approach for Object-Oriented Integration Testing*, Proceedings of World Academy of Science, Engineering and Technology, Volume 16, 2006

LUCRĂRI PUBLICATE

[1]. Anca Elena Iordan, *Pachet de programe software pentru prezentarea metodei « Divide et impera »*, Analele Facultății de Inginerie Hunedoara, 2003, Tomul V, Fascicola 1, pag. 151-158

[2]. Manuela Pănoiu, Caius Pănoiu, Ionel Muscalagiu, Anca Iordan, *Educational software for a comparative study of direct sort methods*, Annals of the Faculty of Engineering Hunedoara, 2004, Tomul II, Fascicola 2, pag. 177-184

[3]. Manuela Pănoiu, Caius Pănoiu, Ionel Muscalagiu, Anca Iordan, *Multimedia educational software for binary tree and their application present*, International Balkan Conference in Informatics, 17-19 noiembrie 2005, Ohrid, Macedonia, pag. 220-227

[4]. Anca Elena Iordan, *Soft educațional pentru prezentarea distribuției binomiale şi a aplicațiilor acesteia la definirea texturilor stohastice***,** A X-a Sesiune de Comunicări Ştiințifice cu participare internațională: Leadership şi management la orizonturile secolului al XXI-lea*,* Sibiu, Romania, 24-26 Noiembrie 2005, Volumul XII, pag. 192-199, ISBN 973-7809-29-7

[5]. Anca Elena Iordan, *Pachet de programe software pentru prezentarea numerelor complexe şi a aplicațiilor acestora în geometrie*, A X-a Sesiune de Comunicări Ştiințifice cu participare internațională: Leadership şi management la orizonturile secolului al XXI-lea*,* Sibiu, Romania, 24-26 Noiembrie 2005, Volumul XII, pag. 200-207, ISBN 973-7809-29-7

[6]. Anca Elena Iordan, Manuela Pănoiu**,** *Multimedia educational software for producing graphs of mathematical functions***,** International Journal of Computers, Communications & Control, 2006, Oradea, Romania, Volumul I, pag. 284-289, ISSN 1841-9836 (ISI Reviste)

[7]. Manuela Pănoiu, Caius Pănoiu, Ionel Muscalagiu, Anca Iordan, *An interactive learning environment for analyze linked list data structures,* International Journal of Computers, Communications & Control, 2006, Oradea, Romania, Volumul I, pag. 355-359, ISSN 1841-9836 (ISI Reviste)

[8]. Anca Elena Iordan, Manuela Pănoiu, *Educational software for the presentation of geometrical elementary locus*, XI-*th* International Conference „Man in the Knowledge Based Organization", 23-25 noiembrie 2006, Sibiu, Romania, ISBN 973-7809-51-3, pag. 136-143

[9]. Anca Elena Iordan, Manuela Pănoiu, *Educational software for exploratory analysis of statistical dates*, XI-*th* International Conference „Man in the Knowledge Based Organization", 23-25 noiembrie 2006, Sibiu, Romania, ISBN 973-7809-51-3, pag. 144-151

[10]. Anca Elena Iordan, Manuela Pănoiu, *Educational software for presentation of the ideal gas laws,* Annals of the Faculty of Engineering Hunedoara, Romania, 2006, Tomul IV, Fascicola 1, ISSN 1584-2665, pag. 87-92 (revista CNCSIS categoria C)

[11]. Anca Elena Iordan, Manuela Pănoiu, *The application of stochastic grammars for generation of fractals,* Annals of the Faculty of Engineering Hunedoara, Romania, 2006, Tomul IV, Fascicola 1, ISSN 1584-2665, pag. 133-140 (revista CNCSIS categoria C)

[12]. Anca Elena Iordan, Manuela Pănoiu, *Dynamical software for the study of Triangle's Geometry*, microCAD 2007 International Scientific Conference, 22-23 martie 2007, Miskolc, Ungaria, pag. 35-40

[13]. Manuela Pănoiu, Anca Iordan, Caius Pănoiu, *Multimedia Educational Software for Study Image Formation in Converging and Diverging Lens*, microCAD 2007 International Scientific Conference, 22-23 martie 2007, Miskolc, Ungaria, pag. 41-46

[14]. Caius Panoiu, Manuela Panoiu, Ionel Muscalagiu, Anca Iordan, *Visual Interactive Environment for study the power electronics using PSCAD-EMTDC simulation program*, Computer Applications in Engineering Education (acceptată pt. publicare în 1.09.2007) (ISI Reviste)

[15]. Anca Iordan, George Savii, Manuela Pănoiu, Caius Pănoiu, *New Software for the Study of the Classic Surfaces from Differential Geometry*, European Computing Conference, Atena, Grecia, 25-27 septembrie 2007, Springer Verlag, pag. 495-504 (ISI Proceedings)

[16]. Anca Iordan, Manuela Pănoiu, Caius Pănoiu, *New Dynamical Methods for the Study of the Euclidian Geometry*, International Conference on Education and Educational Technology, Veneția, Italia, 21-23 Noiembrie 2007, pag. 35-40 (ISI Proceedings)

[17]. Anca Iordan, George Savii, Manuela Pănoiu, Caius Pănoiu, *Development of a Dynamical Software for Teaching Plane Analytical Geometry*, International Conference on Engineering Education, Heraklion, Grecia, 22-24 iulie 2008, pag. 55-60 (ISI Proceedings)

[18]. Manuela Pănoiu, Caius Pănoiu, Anca Iordan, Raluca Rob, *Simulation Results Regarding High Power Loads Balancing,* International Conference on SYSTEMS, Heraklion, Grecia, 22-24 iulie 2008, pag. 614-619 (ISI Proceedings)

[19]. Manuela Pănoiu, Caius Pănoiu, Ioan Sora, Anca Iordan, Raluca Rob, *Using Simulation for study the Possibility of Canceling Load Unbalance of non-sinusoidal High Power three-phase Loads*, WSEAS Transactions on Systems, volum 7, iulie 2008, pag. 699-710 (BDI Reviste)

[20]. Anca Iordan, George Savii, Manuela Pănoiu, Caius Pănoiu, *Development of a dynamical software for doing geometrical constructions*, International Conference on Applied Informatics and Communications, Rodos, Grecia, 20-22 august 2008, pag. 104-109 (ISI Proceedings)

[21]. Anca Iordan, George Savii, Manuela Pănoiu, Caius Pănoiu, *Multimedia Interactive Environment for Study the Plane Analytical Geometry*, WSEAS Transactions on Computers, volum 7, octombrie 2008, pag. 1564-1573 (BDI Reviste)

[22]. Anca Iordan, George Savii, Manuela Pănoiu, Caius Pănoiu, *Visual interactive environment for doing geometrical constructions*, WSEAS Transactions on Computers, volum 8, februarie 2009, pag. 258-268 (BDI Reviste)

ANEXA 1
PREZENTAREA DIAGRAMELOR DE CLASE

Diagrama clasei abstracte *Element2D*

Element2D
tip : String # culoare : Color # stil : BasicStroke # figura : GeneralPath
+ setCuloare (Color) + setStil (BasicStroke) + setare (Color, BasicStroke) + getTip () : String + getCuloare () : Color + getStil () : BasicStroke + getFigura () : GeneralPath + apartine (Punct2D) : boolean + desenare (Graphics2D, int, int)

Diagrama clasei *Punct2D*

Punct2D
- x : double = 0 - y : double = 0 - id : String
+ Punct2D () + Punct2D (double, double) + Punct2D (double, double, String) + Punct2D (Punct2D) + getX () : double + getY () : double + setare (double, double) + setare (double, double, String) + setare (Punct2D) + setX (double) + setY (double) + setId (String) + distanta (Punct2D) : double + coincid (Punct2D) : boolean + coliniare (Punct2D, Punct2D)

Diagrama clasei *Dreapta2D*

Dreapta2D
- A, B, C, m, n : double - ind : int
+ Dreapta2D () + Dreapta2D (Punct2D, Punct2D) + Dreapta2D (Punct2D, double) + Dreapta2D (double, double , double) + Dreapta2D (Vector2D) + Dreapta2D (Punct2D, double, double) + Dreapta2D (Dreapta2D) + getPanta ( ) : double + getOrdonata ( ) : double + getA ( ) : double + getB ( ) : double + getC ( ) : double + getIndicator ( ) : double + vector_director () + setare (Punct2D, Punct2D) + setare (Punct2D, double) + setare (double, double, double) + setare (Vector2D) + setare (Punct2D, double, double) + setare (Dreapta2D) + paralele (Dreapta2D) : boolean + perpendiculare (Dreapta2D) : boolean + paralela (Punct2D) : Dreapta2D + perpendiculara (Punct2D) : Dreapta2D + intersectie (Dreapta2D) : Punct2D + distanta (Punct2D) : double + coincid (Dreapta2D): boolean + concurente (Dreapta2D, Dreapta2D): boolean + unghi() : double + unghi (Dreapta2D) : double + punct1 (int, int) : Punct2D + punct2 (int, int) : Punct2D + punct11 (int, int) : Punct2D + punct22 (int, int) : Punct2D

Diagrama clasei *Semidreapta2D*

Semidreapta2D
- O : Punct2D - u : double - d : Dreapta2D
+ Semidreapta2D ( ) + Semidreapta2D(Punct2D, Punct2D) + Semidreapta2D (Punct2D, double) + Semidreapta2D (Semidreapta2D) + getOrigine ( ) : Punct2D + getSuport ( ) : Dreapta2D + getUnghi ( ) : int + setare (Punct2D, Punct2D) + setare (Punct2D, double) + setare (Semidreapta2D) + coincid (Semidreapta2D) : boolean + semidr_opusa () : Semidreapta2D + unghi (Semidreapta2D) : double

Diagrama clasei *Segment2D*

Segment2D
- A : Punct2D - B : Punct2D - d : Dreapta2D - lungime : double
+ Segment2D ( ) + Segment2D (Punct2D, Punct2D) + Segment2D (double, double, double, double) + Segment2D (Punct2D, double, double) + Segment2D (Segment2D) + getExtremitate1 ( ) : Punct2D + getExtremitate2 ( ) : Punct2D + getSuport ( ) : Dreapta2D + getLungime ( ) : double + setare (Punct2D, Punct2D) + setare (Punct2D, double, double) + setare (Segment2D) + coincid (Segment2D) : boolean + congruente (Segment2D) : boolean + mijloc ( ) : Punct2D + mediatoare () : Dreapta2D

Diagrama clasei *Semiplan2D*

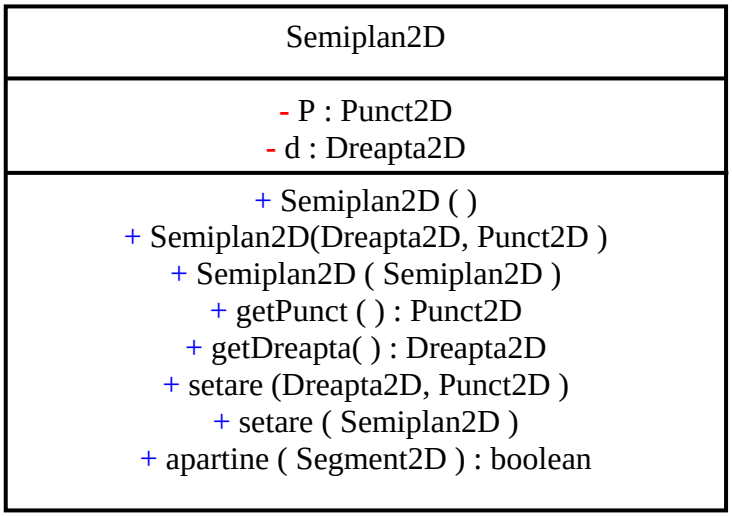


Diagrama clasei *Vector2D*

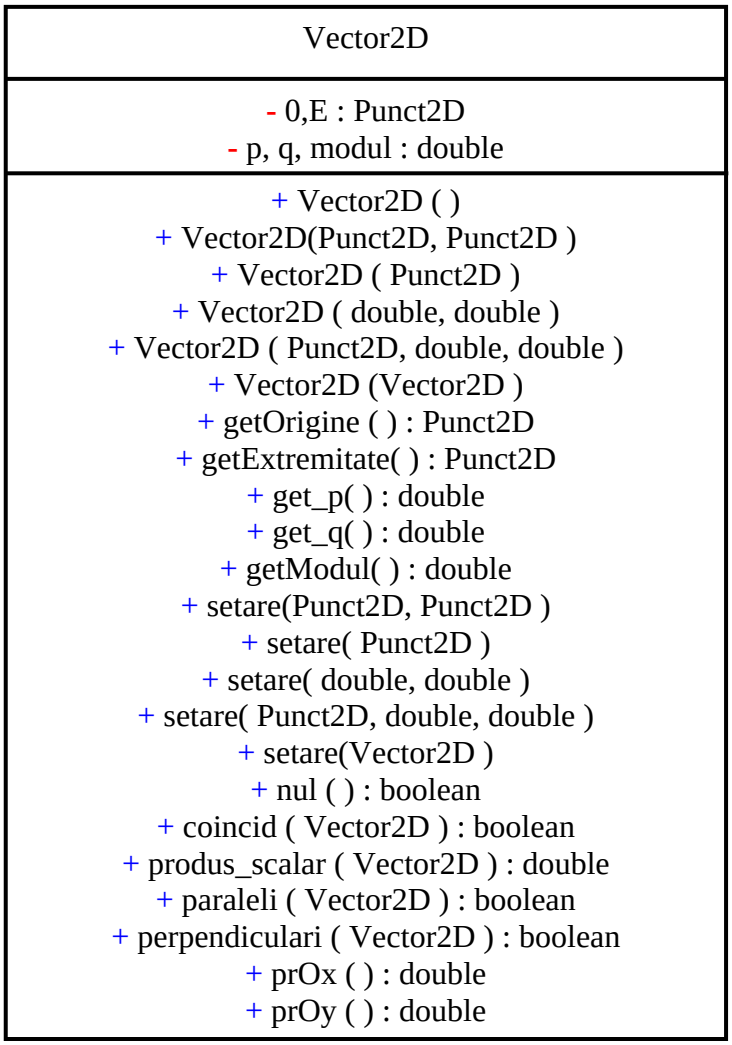


Diagrama clasei *Unghi2D*

Unghi2D
- V : Punct2D - sd1 : Semidreapta2D - sd2 : Semidreapta2D - masura : double
+Unghi2D () +Unghi2D (Punct2D, Semidreapta2D, Semidreapta2D) + Unghi2D (Punct2D, Semidreapta2D, double) + Unghi2D (Punct2D, Punct2D, Punct2D) + Unghi2D (Unghi2D) + getVarf ( ) : Punct2D + getMasura ( ) : double + getSemidreapta1 ( ) : Semidreapta2D + getSemidreapta2 ( ) : Semidreapta2D + setare (Punct2D, Semidreapta2D, Semidreapta2D) + setare (Punct2D, Semidreapta2D, double) + setare(Punct2D, Punct2D, Punct2D) + setare (Unghi2D) + ascutit ( ) : boolean + drept ( ) : boolean + obtuz ( ) : boolean + complementare (Unghi2D) : boolean + suplementare (Unghi2D) : boolean + complement ( ) : double + suplement ( ) : double + bisectoare ( ) : Semidreapta2D

Diagrama clasei *Poligon2D*

Poligon2D
n : int # V[] : Punct2D # sg[] : Segment2D ; # U [] : Unghi2D
+ Poligon2D () + Poligon2D (int, Punct2D [ ]) + Poligon2D (Poligon2D) + getNrVarf ( ) : int + getVarf (int) : Punct2D + getLatura (int) : Segment2D + getUnghi (int) : Unghi2D + setare (int, Punct2D []) + setare (Poligon2D) + perimetru ( ) : double + masura_unghiuri ( ) : double + congruente (Poligon2D) : boolean + convex () : boolean

Diagrama clasei *Poligon_regulat2D*

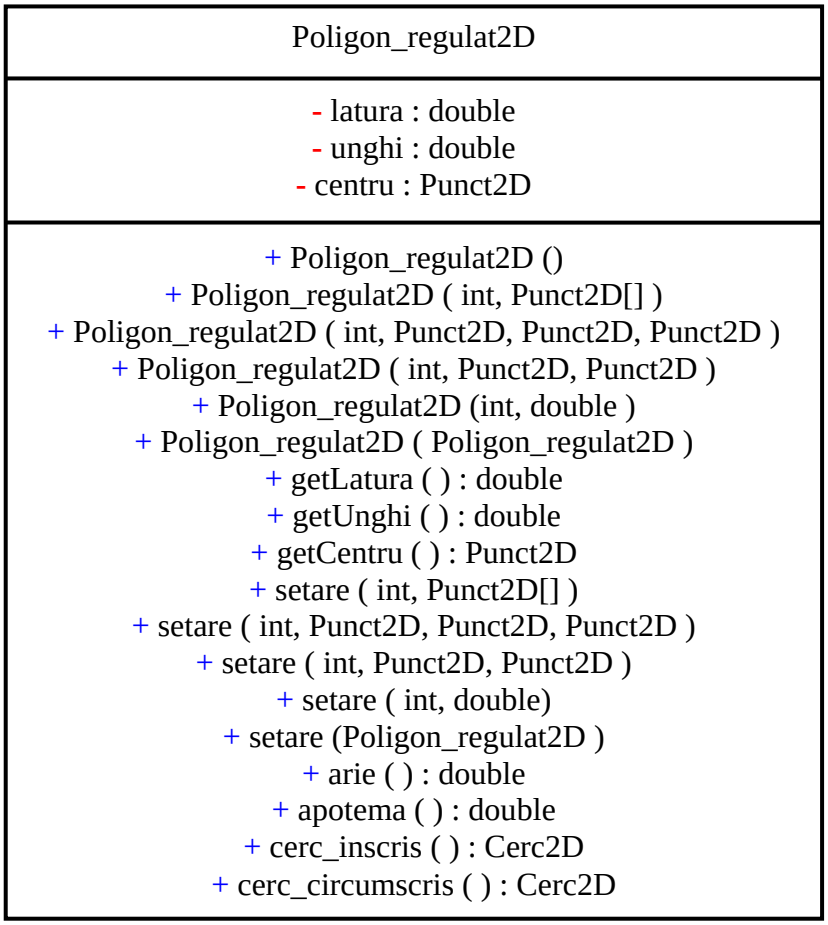


Diagrama clasei *Triunghi_echilateral2D*

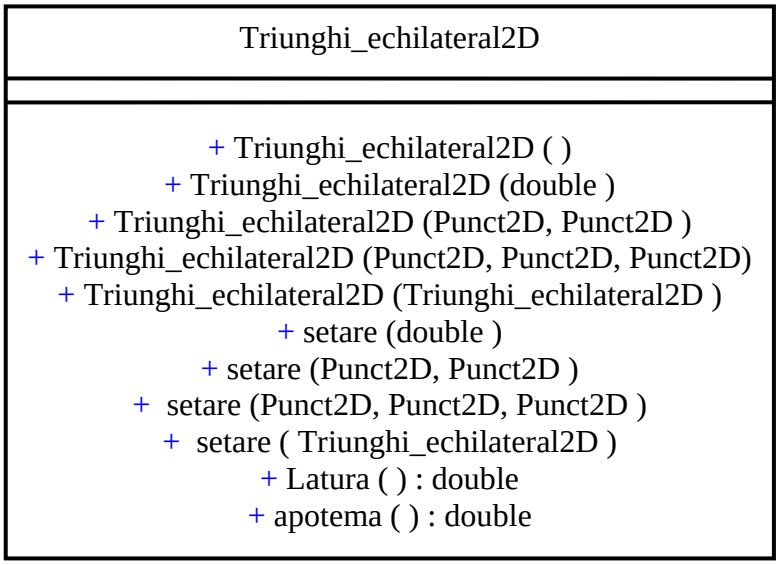


Diagrama clasei *Triunghi2D*

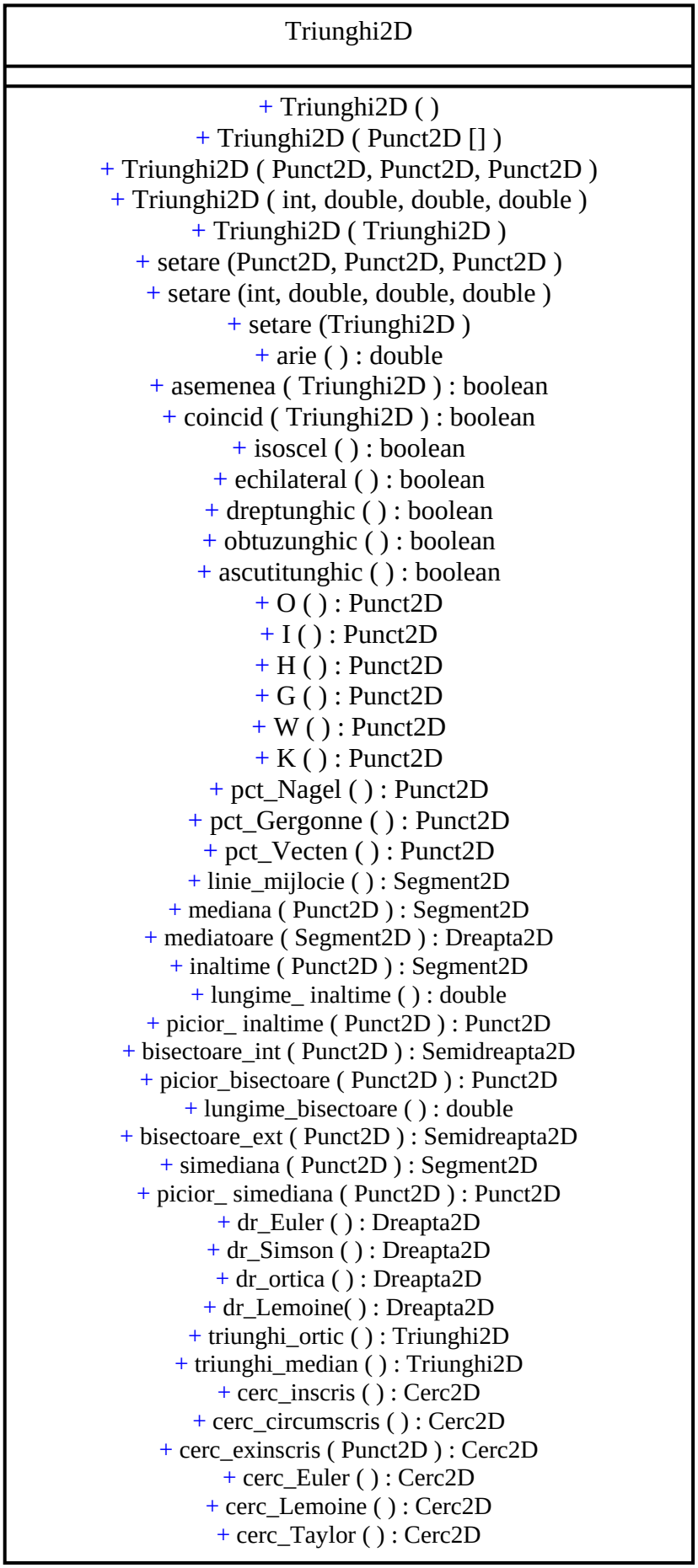


Diagrama clasei *Patrulater2D*

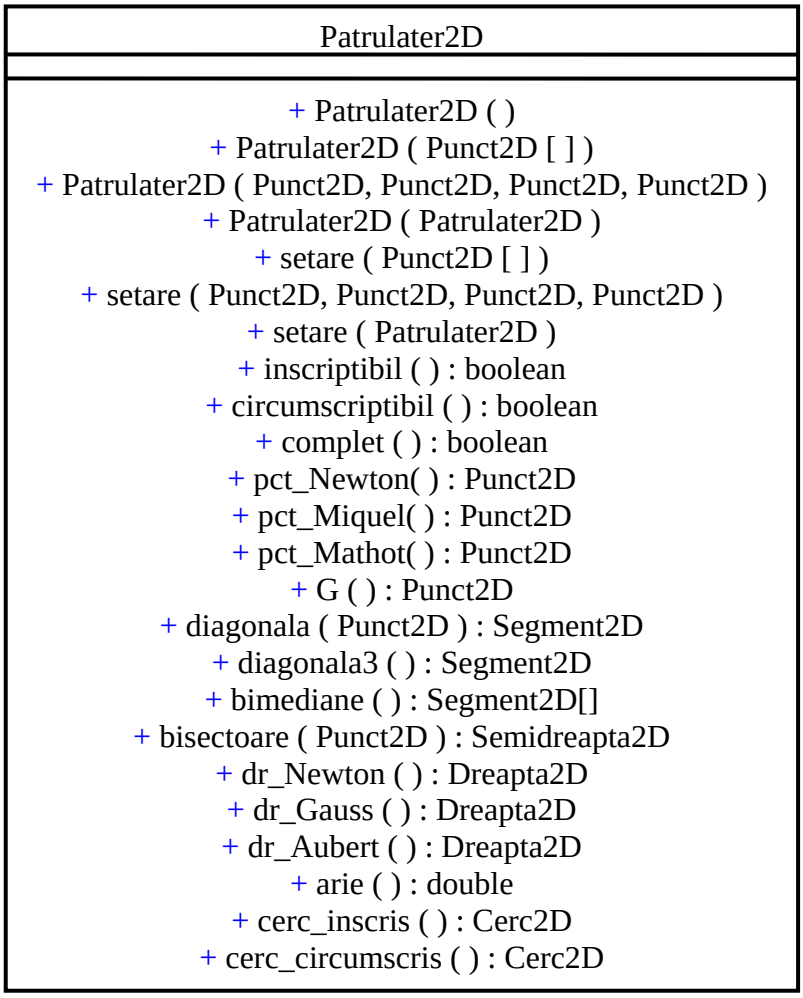


Diagrama clasei *Paralelogram2D*

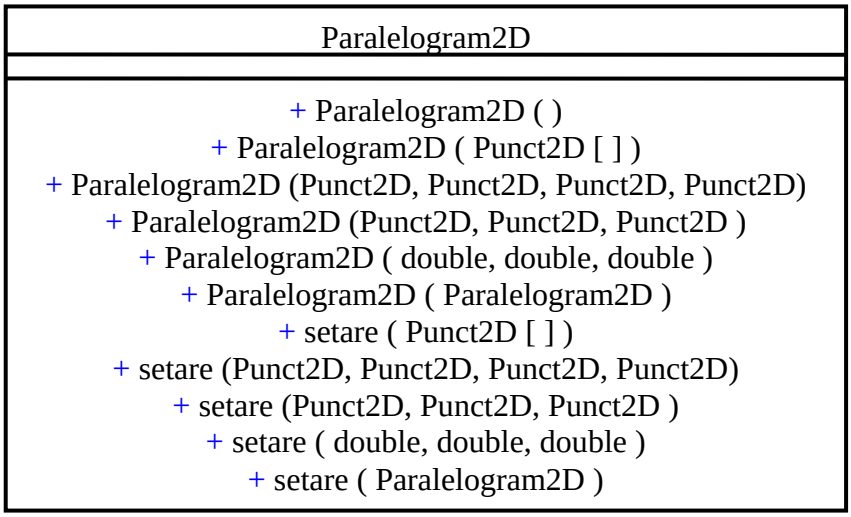


Diagrama clasei *Dreptunghi2D*

Dreptunghi2D
+ Dreptunghi2D ( ) + Dreptunghi2D (Punct2D [ ]) + Dreptunghi2D (Punct2D, Punct2D, Punct2D, Punct2D) + Dreptunghi2D (Punct2D, Punct2D, Punct2D) + Dreptunghi2D (Punct2D, Punct2D, Punct2D, double) + Dreptunghi2D (double, double) + Dreptunghi2D (Dreptunghi2D) + setare (Punct2D [ ]) + setare (Punct2D, Punct2D, Punct2D, Punct2D) + setare (Punct2D, Punct2D, Punct2D) + setare (Punct2D, Punct2D, Punct2D, double) + setare (double, double) + setare (Dreptunghi2D)

Diagrama clasei *Romb2D*

Romb2D
+ Romb2D ( ) + Romb2D (Punct2D [ ]) + Romb2D (Punct2D, Punct2D, Punct2D, Punct2D) + Romb2D (Punct2D, Punct2D, Punct2D) + Romb2D (Punct2D, Punct2D, double) + Romb2D (double, double) + Romb2D (Romb2D) + setare (Punct2D [ ]) + setare (Punct2D, Punct2D, Punct2D, Punct2D) + setare (Punct2D, Punct2D, Punct2D) + setare (Punct2D, Punct2D, double) + setare (double, double) + setare (Romb2D)

Diagrama clasei *Patrat2D*

Patrat2D
+ Patrat2D ( ) + Patrat2D (Punct2D, Punct2D, Punct2D, Punct2D) + Patrat2D (Punct2D, Punct2D, Punct2D) + Patrat2D (double) + Patrat2D (Patrat2D) + setare (Punct2D, Punct2D, Punct2D, Punct2D) + setare (Punct2D, Punct2D, Punct2D) + setare (double) + setare (Patrat2D) + apotema ( ) : double

Diagrama clasei *Trapez2D*

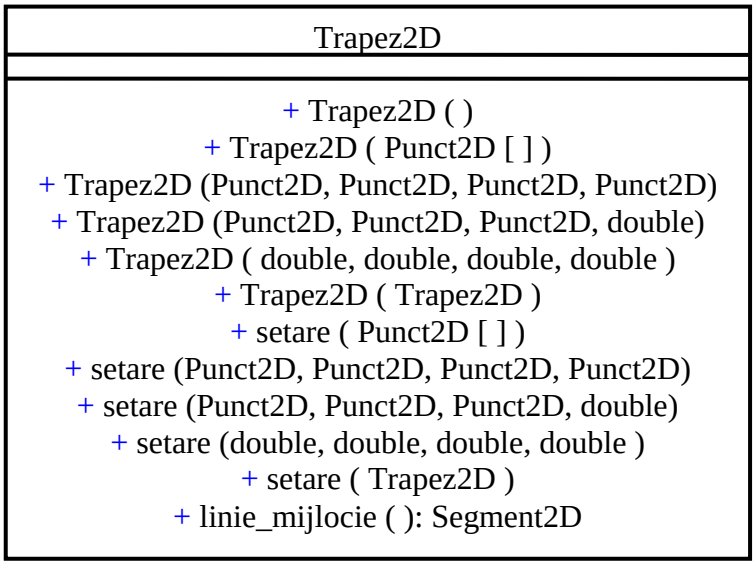


Diagrama clasei abstracte *Conica2D*

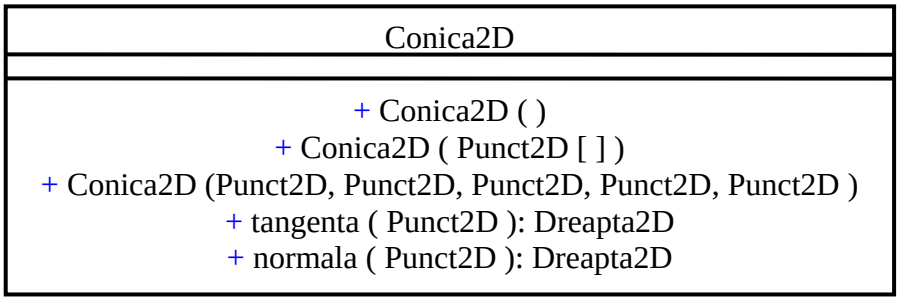


Diagrama clasei *Parabola2D*

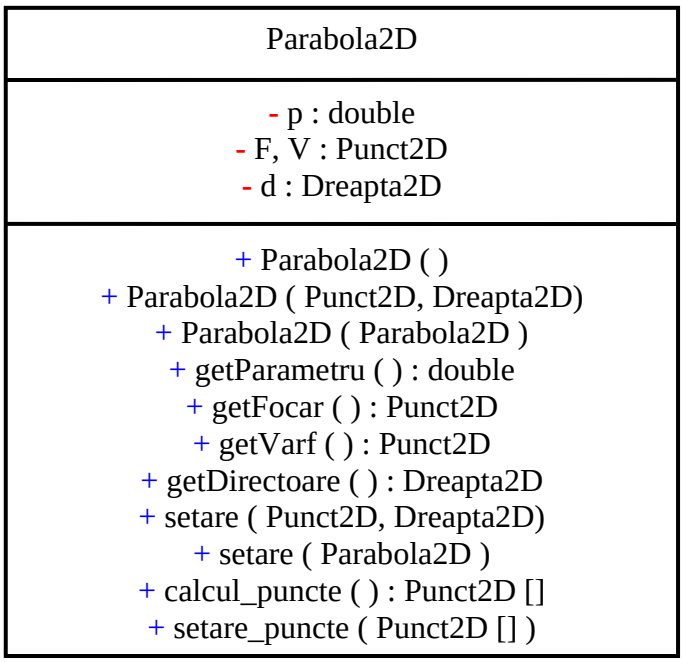


Diagrama clasei *Elipsa2D*

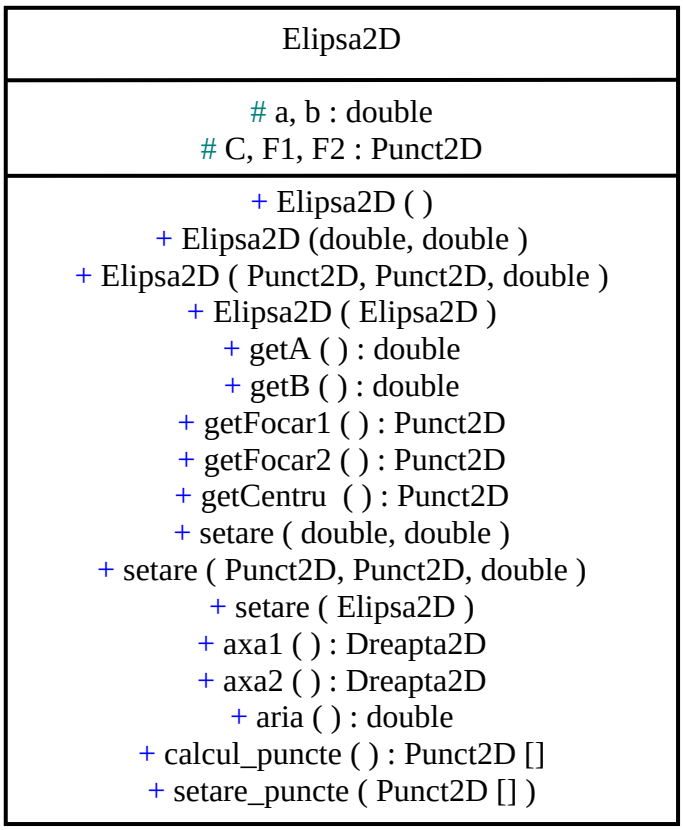


Diagrama clasei *Hiperbola2D*

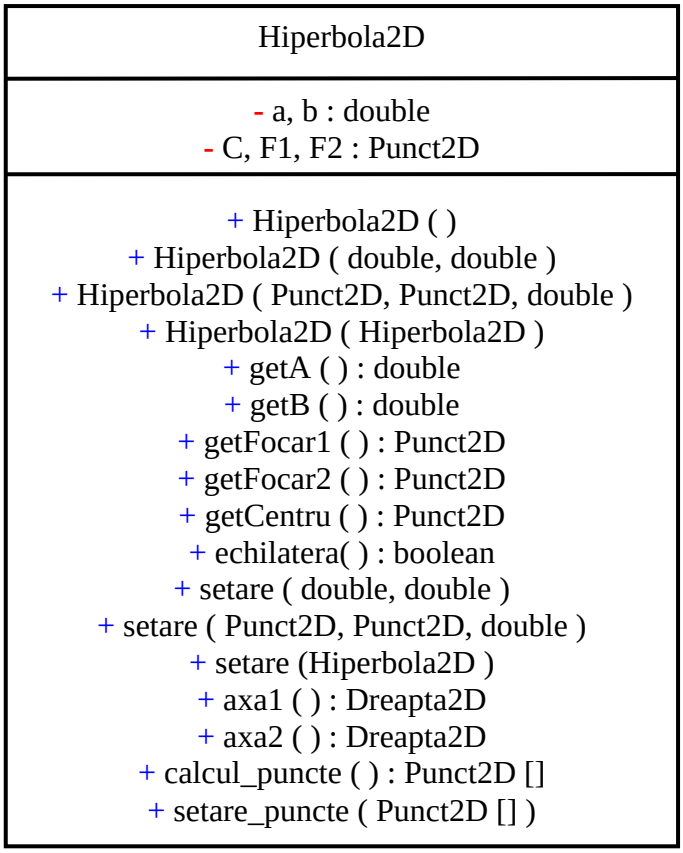


Diagrama clasei *Cerc2D*

Cerc2D
- C : Punct2D - raza : double
+ Cerc2D ( ) + Cerc2D (Punct2D, Punct2D, Punct2D) + Cerc2D (Punct2D, double) + Cerc2D (Cerc2D) + getCentru ( ) : double + getRaza ( ) : double + setare (Punct2D, Punct2D, Punct2D) + setare (Punct2D, double) + setare (Cerc2D) + diametru ( ) : double + aria ( ) : double + lungime ( ) : double + interior(Punct2D) : boolean + exterior(Punct2D) : boolean + tangenta (Dreapta2D) : boolean + punct_tangenta (Dreapta2D) : Punct2D + secanta (Dreapta2D) : boolean + puncte_secanta (Dreapta2D, Punct2D, Punct2D) + tangente (Cerc2D) : boolean + punct_tangenta (Cerc2D) : Punct2D + secante (Cerc2D) : boolean + puncte_secante (Cerc2D, Punct2D, Punct2D) + interior (Cerc2D) : boolean + coarda (Segment2D) : boolean + diametru (Segment2D) : boolean + diametru (Punct2D) : Segment2D + tangente_interioare (Cerc2D) : boolean + tangente_exterioare (Cerc2D) : boolean + coincid (Cerc2D) : boolean

Diagrama clasei *Translatie2D*

Translatie2D
- vector : Segment2D
+ Translatie2D ( ) + Translatie2D (Vector2D) + Translatie2D (Translatie2D) + getVector ( ) : Vector2D + setare (Vector2D) + setare (Translatie2D)

Diagrama clasei abstracte *Izometrie2D*

Izometrie 2D
+ izometrie (Punct2D) : Punct2D + izometrie (Dreapta2D) : Dreapta2D + izometrie (Segment2D) : Segment2D + izometrie (Poligon2D) : Poligon2D + izometrie (Poligon_regulat2D) : Poligon_regulat2D + izometrie (Triunghi2D) : Triunghi2D + izometrie (Triunghi_echilateral2D) : Triunghi_echilateral2D + izometrie (Patrulater2D) : Patrulater2D + izometrie (Paralelogram2D) : Paralelogram2D + izometrie (Dreptunghi2D) : Dreptunghi2D + izometrie (Romb2D) : Romb2D + izometrie (Patrat2D) : Patrat2D + izometrie (Trapez2D) : Trapez2D + izometrie (Cerc2D) : Cerc2D + izometrie (Elipsa2D) : Elipsa2D + izometrie (Hiperbola2D) : Hiperbola2D + izometrie (Parabola2D) : Parabola2D

Diagrama clasei *Simetrie2D*

Simetrie2D
- C : Punct2D - d : Dreapta2D - tip : int
+ Simetrie2D ( ) + Simetrie2D (Punct2D) + Simetrie2D (Dreapta2D) + Simetrie2D (Simetrie2D) + getCentru ( ) : Punct2D + getAxa ( ) : Dreapta2D + setare (Punct2D) + setare (Dreapta2D) + setare (Simetrie2D)

Diagrama clasei *Rotatie2D*

Rotatie2D
- centru : Punct2D - unghi : double
+ Rotatie2D ( ) + Rotatie2D (Punct2D, double) + Rotatie2D (Rotatie2D) + getCentru ( ) : Punct2D + getUnghi ( ) : double + setare (Punct2D, double) + setare (Rotatie2D)

Diagrama clasei *Omotetie2D*

Omotetie2D
- centru : Punct2D - raport : double
+ Omotetie2D ( ) + Omotetie2D (Punct2D, double) + Omotetie2D (Omotetie2D) + getCentru ( ) : Punct2D + getRaport ( ) : double + setare (Punct2D, double) + setare (Omotetie2D) + omotetie (Punct2D) : Punct2D + omotetie (Dreapta2D) : Dreapta2D + omotetie (Segment2D) : Segment2D + omotetie (Poligon2D) : Poligon2D + omotetie (Poligon_regulat2D) : Poligon_regulat2D + omotetie (Triunghi2D) : Triunghi2D + omotetie (Triunghi_echilateral2D) : Triunghi_echilateral2D + omotetie (Patrulater2D) : Patrulater2D + omotetie (Paralelogram2D) : Paralelogram2D + omotetie (Dreptunghi2D) : Dreptunghi2D + omotetie (Romb2D) : Romb2D + omotetie (Patrat2D) : Patrat2D + omotetie (Trapez2D) : Trapez2D + omotetie (Cerc2D) : Cerc2D + omotetie (Elipsa2D) : Elipsa2D + omotetie (Hiperbola2D) : Hiperbola2D + omotetie (Parabola2D) : Parabola2D

Diagrama clasei *Inversiune2D*

Inversiune2D
- pol : Punct2D - putere : double
+ Inversiune2D ( ) + Inversiune2D (Punct2D, double) + Inversiune2D (Inversiune 2D) + getPol ( ) : Punct2D + getPutere ( ) : double + setare (Punct2D, double) + setare (Inversiune2D) + inversiune (Punct2D) : Punct2D + inversiune (Dreapta2D) : Cerc2D + inversiune (Cerc2D) : Dreapta2D + inversiune (Cerc2D) : Cerc2D

Diagrama clasei abstracte *Element3D*

Element3D
tip : String # culoare : Color # stil : BasicStroke # figura : GeneralPath
+ setCuloare (Color) + setStil (BasicStroke) + setare (Color, BasicStroke) + getTip () : String + getCuloare () : Color + getStil () : BasicStroke + getFigura () : GeneralPath + desenare (Graphics2D, int, int) + punct (double, double) : Punct3D + plan_tangent (Punct3D) : Plan3D + normala (Punct3D) : Dreapta3D

Diagrama clasei *Punct3D*

Punct3D
- x : double = 0 - y : double = 0 - z : double = 0 - id : String
+ Punct3D ( ) + Punct3D (double, double, double) + Punct3D (double, double, double, String) + Punct3D (Punct3D) + getX () : double + getY () : double + getZ () : double + getId () : String + setX (double) + setY (double) + setZ (double) + setId (String) + setare (double, double, double) + setare (double, double, double, String) + setare (Punct3D) + distanta (Punct3D) : double + coincid (Punct3D) : boolean + coliniare (Punct3D, Punct3D) + coplanare (Punct3D, Punct3D, Punct3D) + vector_pozitie ( ) : Vector3D

Diagrama clasei *Dreapta3D*

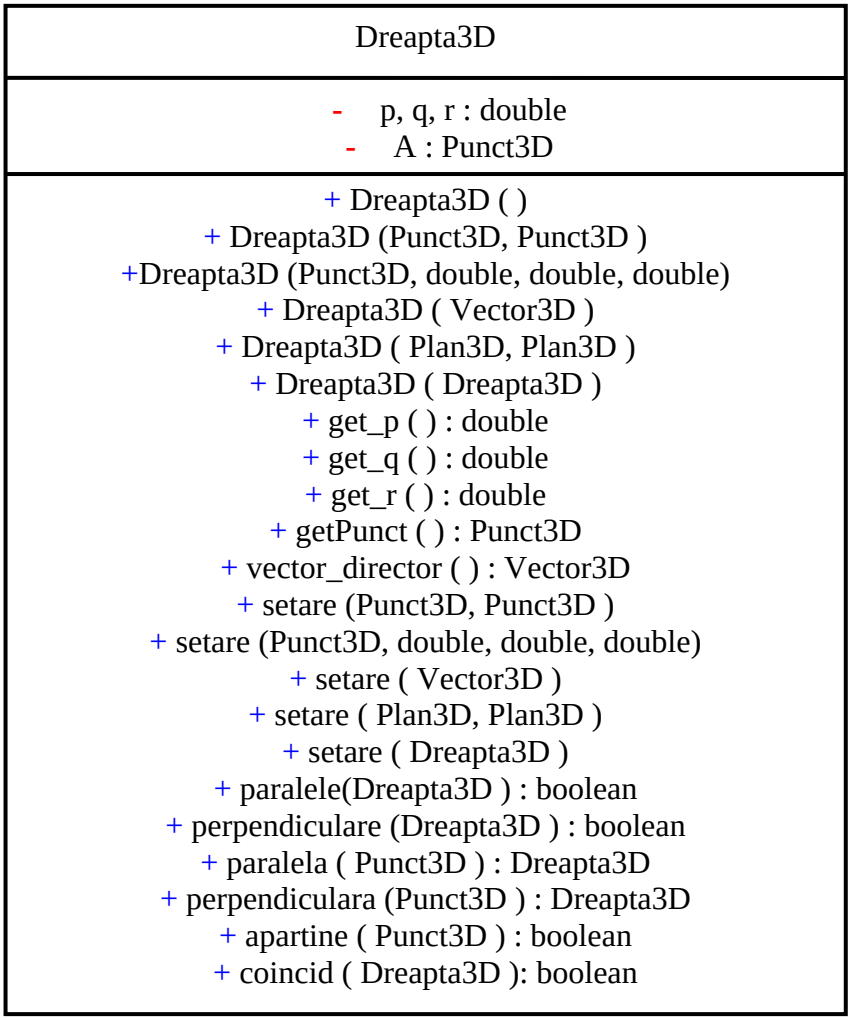


Diagrama clasei *Sfera*

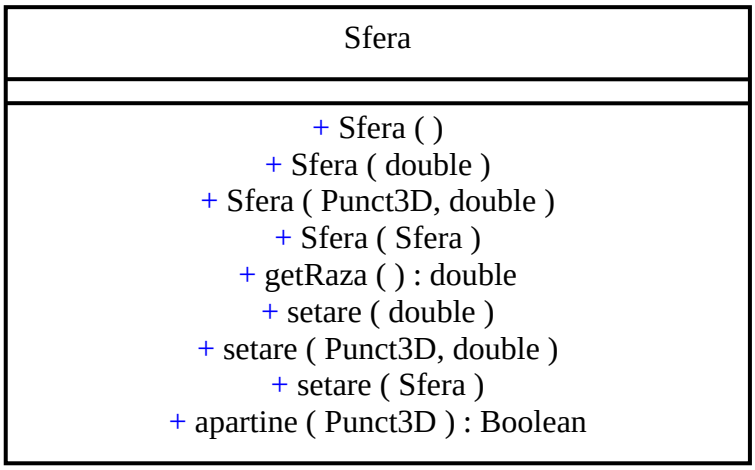


Diagrama clasei *Segment3D*

Segment3D
- A : Punct3D - B : Punct3D - d : Dreapta3D - lungime : double
+ Segment3D ( ) + Segment3D (Punct3D, Punct3D) + Segment3D (Segment3D) + getExtremitate1 ( ) : Punct3D + getExtremitate2 ( ) : Punct3D + getSuport ( ) : Dreapta3D + getLungime ( ) : double + setare (Punct3D, Punct3D) + setare (Segment3D) + coincid (Segment3D) : boolean + congruente (Segment3D) : boolean + apartine (Punct3D) : boolean + mijloc () : Punct3D + mediatoare ( ) : Dreapta3D

Diagrama clasei *Vector3D*

Vector3D
- 0,E : Punct3D - p, q, r, modul : double
+ Vector3D ( ) + Vector3D(Punct3D, Punct3D) + Vector3D (Punct3D) + Vector3D (double, double, double) + Vector3D (Punct3D, double, double, double) + Vector3D (Vector3D) + getOrigine ( ) : Punct3D + getExtremitate( ) : Punct3D + get_p( ) : double + get_q( ) : double + get_r( ) : double + getModul( ) : double + setare (Punct3D, Punct3D) + setare (Punct3D) + setare (double, double, double) + setare (Punct3D, double, double, double) + setare (Vector3D) + nul ( ) : boolean + coincid (Vector3D) : boolean + produs_scalar (Vector3D) : double + paraleli (Vector3D) : boolean + perpendiculari (Vector3D) : boolean + prOx ( ) : double + prOy ( ) : double + prOz ( ) : double

Diagrama clasei *Plan3D*

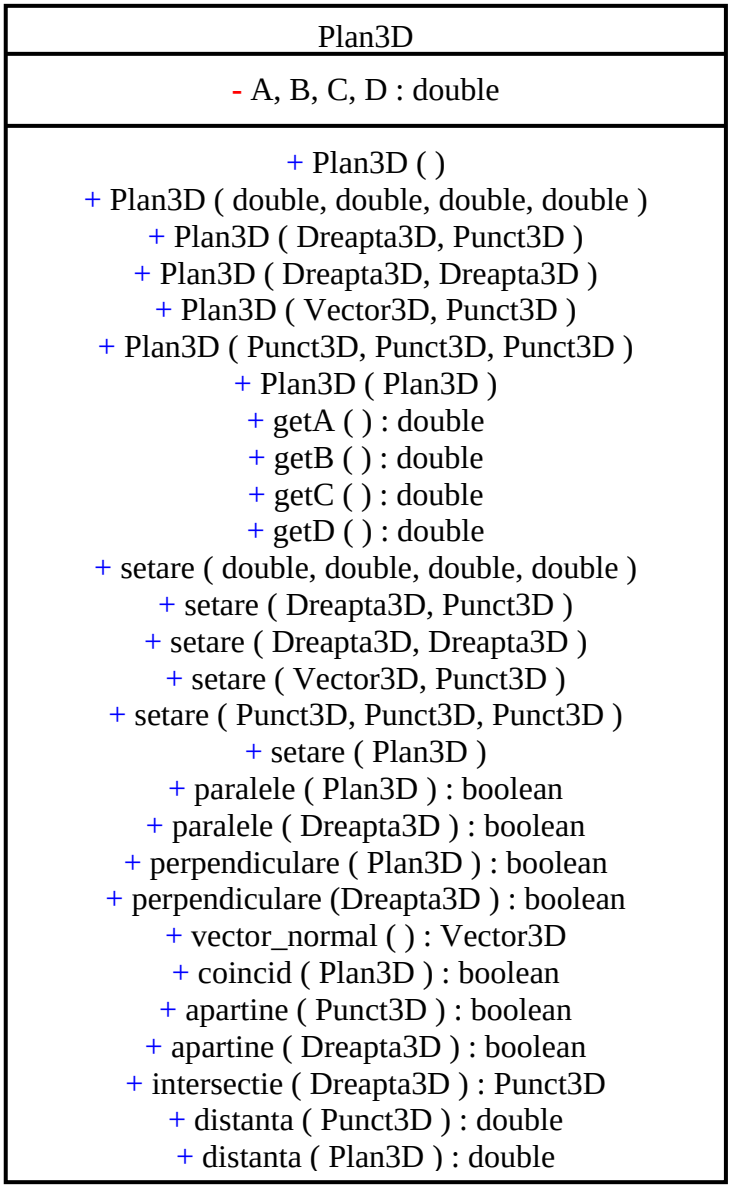


Diagrama clasei *Elicoid*

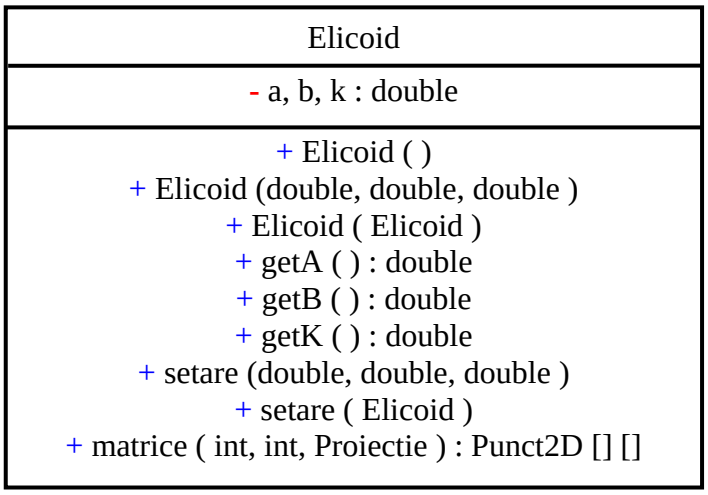


Diagrama clasei *Elipsoid*

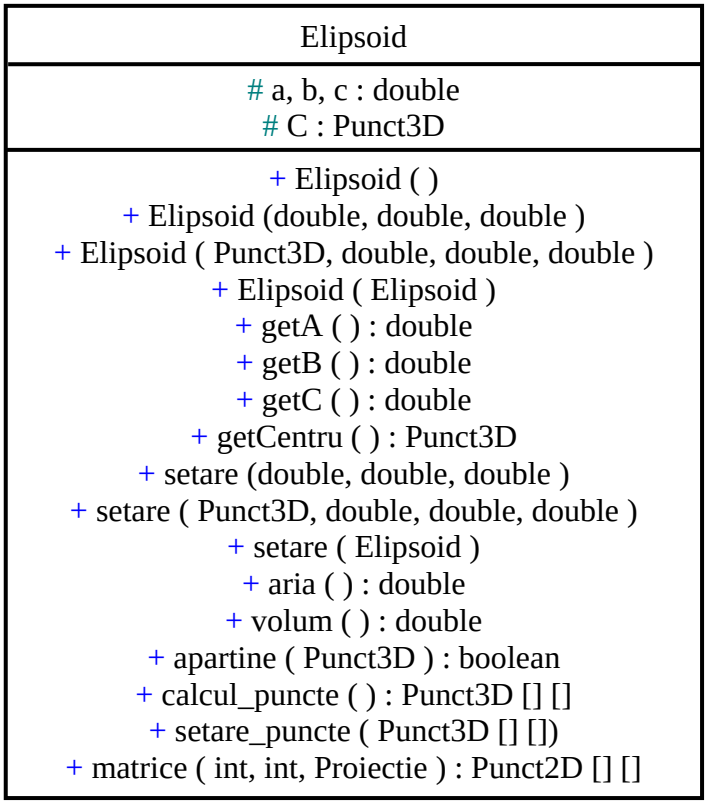


Diagrama clasei *Hiperboloid*

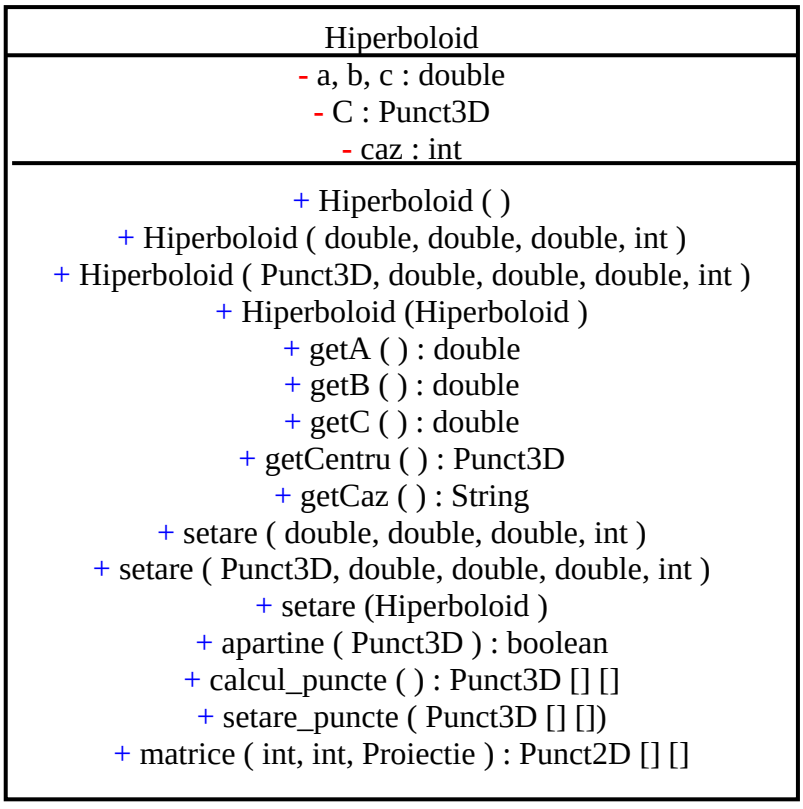


Diagrama clasei *Paraboloid*

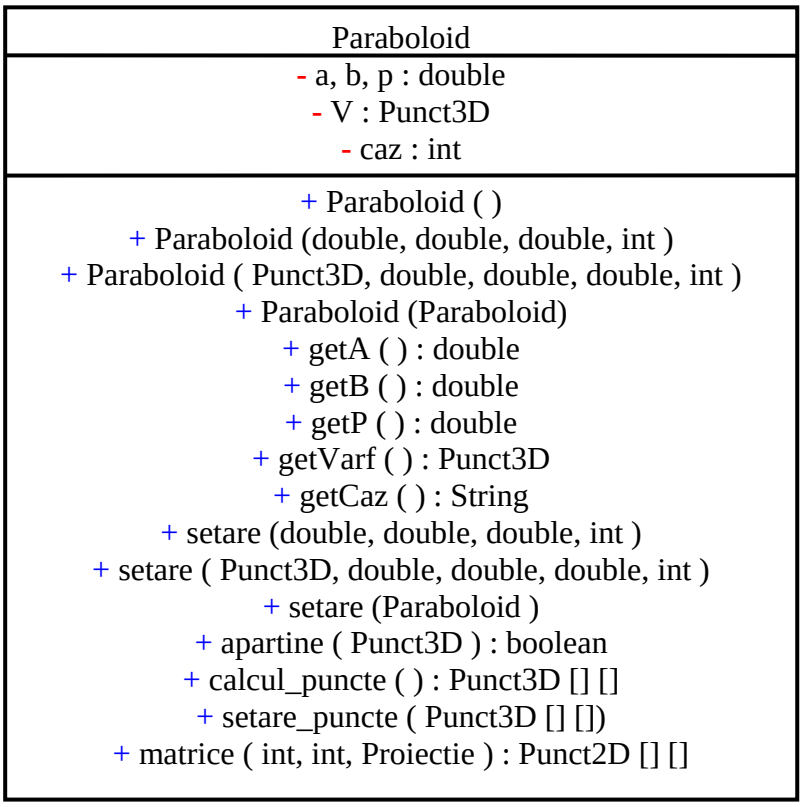


Diagrama clasei *Pseudosfera*

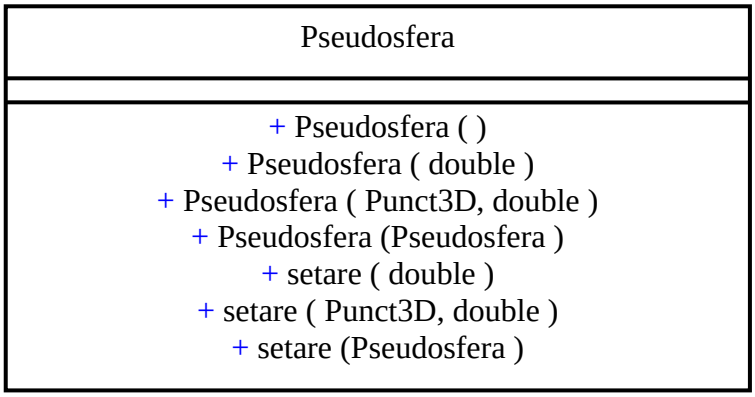


Diagrama clasei abstracte *Izometrie3D*

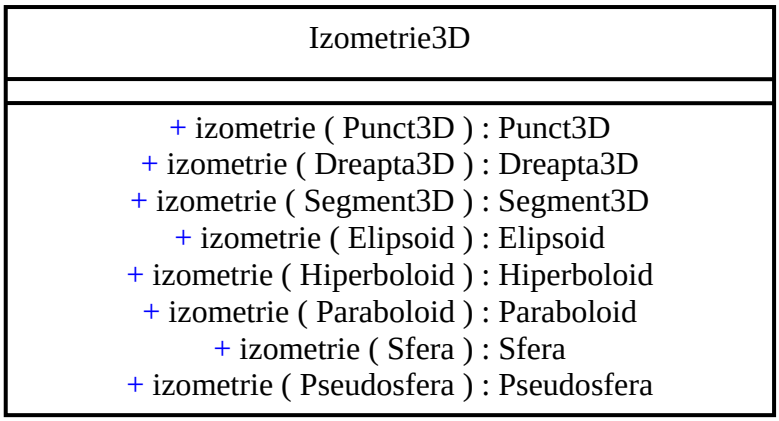


Diagrama clasei *Simetrie3D*

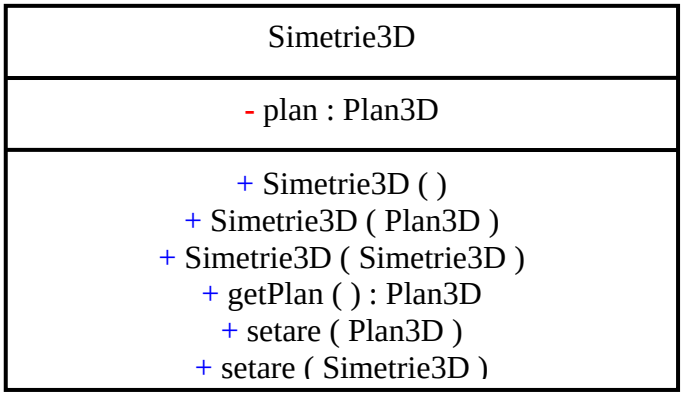


Diagrama clasei *Rotatie3D*

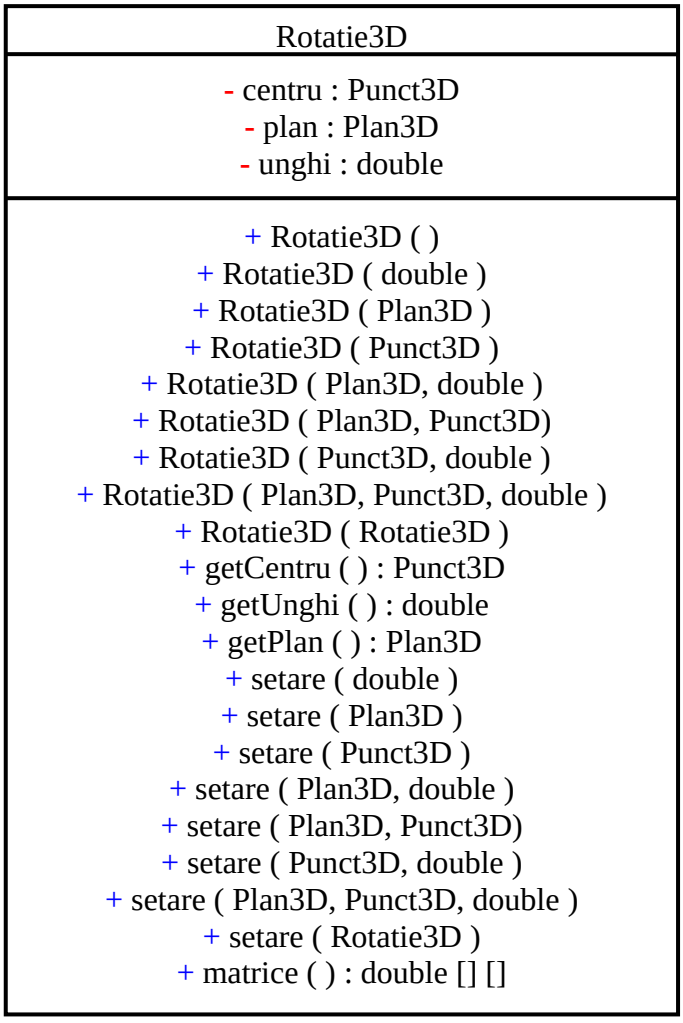


Diagrama clasei *Translatie3D*

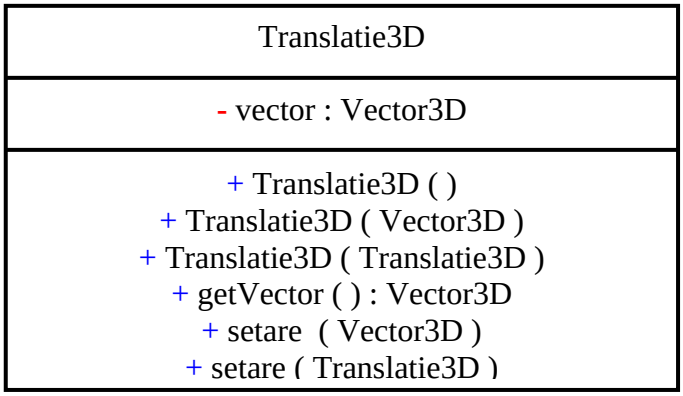
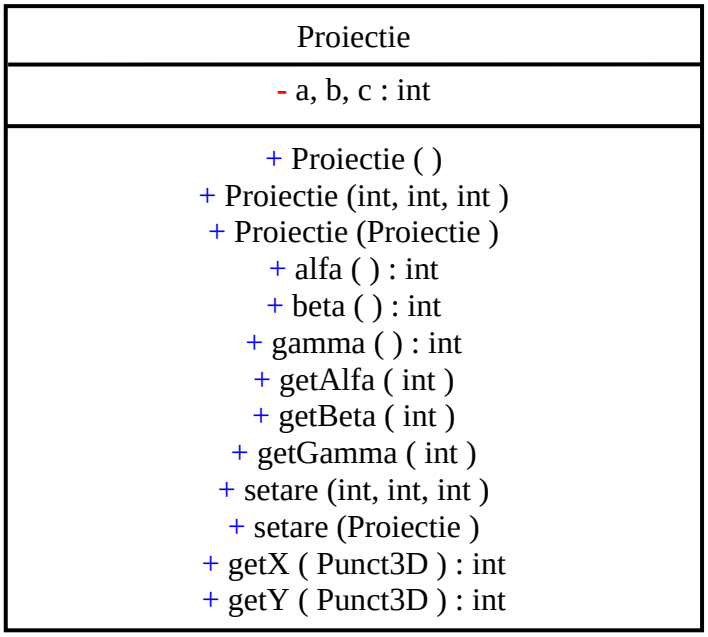


Diagrama clasei *Proiectie*



ANEXA 2

Clasa **Dreapta2D**

```
public class Dreapta2D extends Element2D {
        private double A,B,C;  /coeficientii din ecuatia unei drepte
        private double m,n;  //panta unei drepte si ordonata la origine
        private int ind;
        //metode
          //constructori
        public Dreapta2D() {
                A=0;
                B=1;
                C=0;
                m=0;
                n=0;
                ind=2;
                tip="Dreapta";
        }
        public Dreapta2D(Punct2D A1,Punct2D B1){
                double dx,dy;
                dx=A1.getX()-B1.getX();
                dy=A1.getY()-B1.getY();
                if (dx==0) {
                        ind=3;  //dreapta verticala
                        A=1;
                        B=0;
                        C=-A1.getX();
                        m=Double.POSITIVE_INFINITY;
                        n=Double.POSITIVE_INFINITY;
                }
                else if (dy==0){
                        ind=2;  //dreapta orizontala
                        m=0;
                        n=A1.getY();
                        A=0;
                        B=1;
                        C=-n;
                }
                else {
                        ind=1;  //dreapta oblica
                        m=dy/dx;
                        n=A1.getY()-m*A1.getX();
                        A=-m;
                        B=1;
                        C=-n;
                }
                tip="Dreapta";
        }
        public Dreapta2D(Punct2D A1, double m1){
           if (m1==Double.POSITIVE_INFINITY) {
```

```
                ind=3;  //dreapta verticala
                A=1;
                B=0;
                C=-A1.getX();
                m=Double.POSITIVE_INFINITY;
                n=Double.POSITIVE_INFINITY;
        }
        else if (m1==0){
                ind=2;  //dreapta orizontala
                m=0;
                n=A1.getY();
                A=0;
                B=1;
                C=-n;
        }
        else {
                ind=1;  //dreapta oblica
                m=m1;
                n=A1.getY()-m*A1.getX();
                A=-m;
                B=1;
                C=-n;
        }
        tip="Dreapta";
}
public Dreapta2D(double a,double b,double c){
        A=a;
        B=b;
        C=c;
        if (A==0){  //dreapta orizontala
                m=0;
                n=-C/B;
                ind=2;
        }
        else if (B==0){  //dreapta verticala
           ind=3;
           m=Double.POSITIVE_INFINITY;
           n=Double.POSITIVE_INFINITY;
        }
        else {  //dreapta oblica
                ind=1;
                m=-A/B;
                n=-C/B;
        }
        tip="Dreapta";
}
public Dreapta2D(Vector2D v) {
        if (v.get_p()==0) {  //dreapta verticala
           ind=3;
           m=Double.POSITIVE_INFINITY;
           n=Double.POSITIVE_INFINITY;
           A=1;
           B=0;
           C=-v.getOrigine().getX();
        }
        else if (v.get_q()==0) {  //dreapta orizontala
             ind=2;
```

```
                A=0;
                B=1;
                C=-v.getOrigine().getY();
                m=0;
                n=-C/B;
            }
            else {
                ind=1;
                A=-v.get_q()/v.get_p();
                B=1;
                C=-v.getOrigine().getY()+v.get_q()*v.getOrigine().getX()/v.get_p();
                m=-A/B;
                n=-C/B;
            }
    }
    public Dreapta2D(Punct2D P, double p, double q) {
            if (p==0) {  //dreapta verticala
                ind=3;
                m=Double.POSITIVE_INFINITY;
                n=Double.POSITIVE_INFINITY;
                A=1;
                B=0;
                C=-P.getX();
            }
            else if (q==0) {  //dreapta orizontala
                ind=2;
                A=0;
                B=1;
                C=-P.getY();
                m=0;
                n=-C/B;
            }
            else {
                ind=1;
                A=-q/p;
                B=1;
                C=-P.getY()+q*P.getX()/p;
                m=-A/B;
                n=-C/B;
            }
    }
    public Dreapta2D(Dreapta2D d){
            A=d.A;
            B=d.B;
            C=d.C;
            m=d.m;
            n=d.n;
            ind=d.ind;
            tip="Dreapta";
   culoare=d.culoare;
    stil=d.stil;
    }
      //functii de acces
    public double getPanta(){
            return m;
    }
    public double getOrdonata(){
```

```
            return n;
    }
    public double getA(){
            return A;
    }
    public double getB(){
            return B;
    }
    public double getC(){
            return C;
    }
    public int getIndicator(){
            return ind;
    }
    public Vector2D vector_director() {
            Punct2D P=new Punct2D();
            double p,q;
            if (ind==3) {   //dreapta verticala
                    P.setare(-C,0);
                    p=0;
                    q=1;
            }
            else if (ind==2) {   //dreapta orizontala
                    P.setare(0,-C);
                    p=1;
                    q=0;
            }
            else {
                    P.setare(0,-C/B);
                    p=1;
                    q=-A/B;
            }
            return new Vector2D(P,p,q);
    }
      //modificatori
    public void setare(Punct2D A1,Punct2D B1){
            double dx,dy;
            dx=A1.getX()-B1.getX();
            dy=A1.getY()-B1.getY();
            if (dx==0) {
                    ind=3;  //dreapta verticala
                    A=1;
                    B=0;
                    C=-A1.getX();
                    m=Double.POSITIVE_INFINITY;
                    n=Double.POSITIVE_INFINITY;
            }
            else if (dy==0){
                    ind=2;  //dreapta orizontala
                    m=0;
                    n=A1.getY();
                    A=0;
                    B=1;
                    C=-n;
            }
            else {
                    ind=1;  //dreapta oblica
```

```
                m=dy/dx;
                n=A1.getY()-m*A1.getX();
                A=-m;
                B=1;
                C=-n;
        }
}
public void setare(Punct2D A1, double m1){
   if (m1==Double.POSITIVE_INFINITY) {
                ind=3;  //dreapta verticala
                A=1;
                B=0;
                C=-A1.getX();
                m=Double.POSITIVE_INFINITY;
                n=Double.POSITIVE_INFINITY;
        }
        else if (m1==0){
                ind=2;  //dreapta orizontala
                m=0;
                n=A1.getY();
                A=0;
                B=1;
                C=-n;
        }
        else {
                ind=1;  //dreapta oblica
                m=m1;
                n=A1.getY()-m*A1.getX();
                A=-m;
                B=1;
                C=-n;
        }
}
public void setare(double a,double b,double c){
        A=a;
        B=b;
        C=c;
        if (A==0){  //dreapta orizontala
                m=0;
                n=-C/B;
                ind=2;
        }
        else if (B==0){  //dreapta verticala
           ind=3;
           m=Double.POSITIVE_INFINITY;
           n=Double.POSITIVE_INFINITY;
        }
        else {   //dreapta oblica
                ind=1;
                m=-A/B;
                n=-C/B;
        }
}
public void setare(Vector2D v) {
        if (v.get_p()==0) {   //dreapta verticala
           ind=3;
           m=Double.POSITIVE_INFINITY;
```

```
                n=Double.POSITIVE_INFINITY;
                A=1;
                B=0;
                C=-v.getOrigine().getX();
            }
            else if (v.get_q()==0) {  //dreapta orizontala
                ind=2;
                A=0;
                B=1;
                C=-v.getOrigine().getY();
                m=0;
                n=-C/B;
            }
            else {
                ind=1;
                A=-v.get_q()/v.get_p();
                B=1;
                C=-v.getOrigine().getY()+v.get_q()*v.getOrigine().getX()/v.get_p();
                m=-A/B;
                n=-C/B;
            }
    }
    public void setare(Punct2D P, double p, double q) {
            if (p==0) {  //dreapta verticala
                ind=3;
                m=Double.POSITIVE_INFINITY;
                n=Double.POSITIVE_INFINITY;
                A=1;
                B=0;
                C=-P.getX();
            }
            else if (q==0) {  //dreapta orizontala
                ind=2;
                A=0;
                B=1;
                C=-P.getY();
                m=0;
                n=-C/B;
            }
            else {
                ind=1;
                A=-q/p;
                B=1;
                C=-P.getY()+q*P.getX()/p;
                m=-A/B;
                n=-C/B;
            }
    }
    public void setare(Dreapta2D d){
            A=d.A;
            B=d.B;
            C=d.C;
            m=d.m;
            n=d.n;
            ind=d.ind;
            tip="dreapta";
            culoare=d.culoare;
```

```
            stil=d.stil;
    }
     //verifica daca doua drepte sunt paralele
    public boolean paralele(Dreapta2D D){
            return m==D.m;
    }
     //verifica daca doua drepte sunt perpendicuare
    public boolean perpendiculare(Dreapta2D D){
            boolean cod=false;
            if (ind==3 && D.ind==2) cod=true;
            if (ind==2 && D.ind==3) cod=true;
            if (ind==1 && D.ind==1 && m*D.m==-1) cod=true;
            return cod;
    }
     //determina paralela la dreapta data ce trece printr-un punct dat
    public Dreapta2D paralela(Punct2D p){
       Dreapta2D d=new Dreapta2D(p,m);
       return d;
    }
     //determina perpendiculara pe dreapta data ce trece printr-un punct dat
    public Dreapta2D perpendiculara(Punct2D p){
            double m1;
            if(m==Double.POSITIVE_INFINITY) m1=0;
            else if (m==0) m1=Double.POSITIVE_INFINITY;
            else m1=-1/m;
            Dreapta2D d=new Dreapta2D(p,m1);
            return d;
    }
     //verifica daca punctul transmis ca parametru apartine dreptei curente
    public boolean apartine(Punct2D p){
            boolean cod=false;
            double x,y;
            if (ind==2)
               if ( p.getY()==n  ) cod=true;
            if (ind==3)
               if ( p.getX()>=-C-2 && p.getX()<=-C+2 ) cod=true;
            if (ind==1) {
            int i=-3,j;
            while (i<=3 && !cod) {
                    x=p.getX()+i;
                    j=-3;
                    while (j<=3 && !cod){
                            y=p.getY()+j;
                            if (A*x+B*y+C>=-4 && A*x+B*y+C<=4) cod=true;
                            j++;
                    }
                    i++;
            }
            }
            return cod;
    }
      //returneaza punctul de intersectie a doua drepte neparalele
    public Punct2D intersectie(Dreapta2D d){
            Punct2D p=new Punct2D();
            double dx,dy;
            dx=-C*d.B+d.C*B;
            dy=A*d.B-d.A*B;
```

```
            p.setX(dx/dy);
            if (ind==3)  p.setY(d.m*p.getX()+d.n);
            else p.setY(m*p.getX()+n);
            return p;
}
  //returneaza distanta de la punctul dat ca parametru la dreapta curenta
public double distanta(Punct2D P){
            Dreapta2D d=new Dreapta2D(perpendiculara(P));
            Punct2D M=new Punct2D(intersectie(d).getX(),intersectie(d).getY());
            double dist=P.distanta(M);
            return dist;
}
  //verifica coincidenta a doua drepte
public boolean coincid(Dreapta2D d){
            boolean cod;
            if (ind!=d.ind) cod=false;
            else if (ind==3)//drepte verticale
              if (C/A==d.C/d.A) cod=true;
              else cod=false;
              else if (ind==2)//drepte orizontale
                if (C/B==d.C/d.B) cod=true;
                else cod=false;
                else //drepte oblice
                 if (m==d.m&&n==d.n) cod=true;
                 else cod=false;
            return cod;
}
  // verifica concurenta a trei drepte
public boolean concurente(Dreapta2D d1, Dreapta2D d2){
    boolean cod;
    if (A*d1.B-d1.A*B==0&&A*d2.B-d2.A*B==0&&d1.A*d2.B-d2.A*d1.B==0)  cod=false;
    else
   if (A*d1.B*d2.C+d1.A*d2.B*C+d2.A*B*d1.C-d2.A*d1.B*C-A*d2.B*d1.C-d1.A*B*d2.C!=0)
    cod=false;
    else cod=true;
    return cod;
}
  //determina unghiul pe care il face dreapta cu axa Ox
public double unghi(){
            double u;
            if (ind==2) u=0;
            else if (ind==3) u=90;
            else //u=180*Math.atan(m)/Math.PI;
            u=Math.toDegrees(Math.atan(m));
            return u;
}
  //determina unghiul dintre doua drepte
public double unghi(Dreapta2D d){
            double u=Math.abs(unghi()-d.unghi());
            if (u>90) u=180-u;
            return u;
}
  //returneaza o extremitate a segmentului de dreapta ce se incadreaza intr-un dreptunghi
public Punct2D punct1(int a,int b){
            Punct2D D=new Punct2D();
            if (B==0) {
                        D.setX(-C/A);
```

```
                D.setY(0);
            }
            else {
                D.setX(0);
                D.setY(-C/B);
            }
            return D;
        }
          //returneaza cealalta extremitate a segmentului de dreapta ce se incadreaza intr-un dreptunghi
        public Punct2D punct2(int a,int b){
            Punct2D D=new Punct2D();
            if (B==0) {
                D.setX(-C/A);
                D.setY(b);
            }
            else {
                D.setX(a);
                D.setY((-A*a-C)/B);
            }
            return D;
        }
        //returneaza o extremitate a segmentului de dreapta ce se incadreaza intr-un dreptunghi de lungimi 2a si 2b
        public Punct2D punct11(int a,int b){
            Punct2D D=new Punct2D();
            if (B==0) {
                D.setX(-C/A);
                D.setY(-b);
            }
            else {
                D.setX(-a);
                D.setY((a*A-C)/B);
            }
            return D;
        }
          //returneaza cealalta extremitate a segmentului de dreapta ce se incadreaza intr-un dreptunghi de lungimi 2a si 2b
        public Punct2D punct22(int a,int b){
            Punct2D D=new Punct2D();
            if (B==0) {
                D.setX(-C/A);
                D.setY(b);
            }
            else {
                D.setX(a);
                D.setY((-A*a-C)/B);
            }
            return D;
        }
        //metoda de desenare a elementului geometric
        public void desenare(Graphics2D g, int latime, int inaltime) {
            figura=new GeneralPath();
            int a,b,xe,ye;
            a=(int)(latime/2);
            b=(int)(inaltime/2);
            xe=Desen3D.xecran(latime,a,punct11(latime,inaltime).getX());
            ye=Desen3D.yecran(inaltime,b,punct11(latime,inaltime).getY());
```

```
                figura.moveTo(xe,ye);
                xe=Desen3D.xecran(latime,a,punct22(latime,inaltime).getX());
                ye=Desen3D.yecran(inaltime,b,punct22(latime,inaltime).getY());
                figura.lineTo(xe,ye);
                g.setColor(culoare);
                g.setStroke(stil);
                g.draw(figura);
        }
        }
```

Clasa **Triunghi2D**

```
public class Triunghi2D extends Poligon2D {
        //metode
          //constructori
        public Triunghi2D() {
                nr=3;
                V=new Punct2D[3];
                V[0]=new Punct2D(10,0);
                V[1]=new Punct2D(0,30);
                V[2]=new Punct2D(-10,0);
                for (int i=0;i<nr-1;i++) sg[i]=new Segment2D(V[i],V[i+1]);
                sg[nr-1]=new Segment2D(V[nr-1],V[0]);
                for (int i=1;i<nr-1;i++) U[i]=new Unghi2D(V[i],V[i-1],V[i+1]);
                U[0]=new Unghi2D(V[0],V[nr-1],V[1]);
                U[nr-1]=new Unghi2D(V[nr-1],V[nr-2],V[0]);
                tip="Triunghi";
        }
        public Triunghi2D(Punct2D[] P) {
                super(3,P);
                tip="Triunghi";
        }
        public Triunghi2D(Punct2D P1, Punct2D P2, Punct2D P3) {
                nr=3;
                V=new Punct2D[3];
                V[0]=new Punct2D(P1);
                V[1]=new Punct2D(P2);
                V[2]=new Punct2D(P3);
                for (int i=0;i<nr-1;i++) sg[i]=new Segment2D(V[i],V[i+1]);
                sg[nr-1]=new Segment2D(V[nr-1],V[0]);
                for (int i=1;i<nr-1;i++) U[i]=new Unghi2D(V[i],V[i-1],V[i+1]);
                U[0]=new Unghi2D(V[0],V[nr-1],V[1]);
                U[nr-1]=new Unghi2D(V[nr-1],V[nr-2],V[0]);
                tip="Triunghi";
        }
        public Triunghi2D(int caz, double a, double b, double c) {
                nr=3;
                V=new Punct2D[3];
                if (caz==1) {   //cazul LLL
                   V[0]=new Punct2D(0,0);
                   V[1]=new Punct2D(a,0);
                   double u=(a*a+c*c-b*b)/(2*a*c);
                   double u1=Math.acos(u);
                   V[2]=new Punct2D(c*u, c*Math.sin(u1));
                }
```

```
            else if (caz==2) {   //cazul LUL
               V[0]=new Punct2D(0,0);
               V[1]=new Punct2D(a,0);
               double u=180-b;
            V[2]=new Punct2D(a+c*Math.cos(Math.toRadians(u)), c*Math.sin(Math.toRadians(u)));
            }
            else {   //cazul ULU
               V[0]=new Punct2D(0,0);
               V[1]=new Punct2D(b,0);
               double lat=b*Math.sin(Math.toRadians(c))/Math.sin(Math.toRadians(180-a-c));
             V[2]=new Punct2D(lat*Math.cos(Math.toRadians(a)),lat*Math.sin(Math.toRadians(a)));
            }
            for (int i=0;i<nr-1;i++) sg[i]=new Segment2D(V[i],V[i+1]);
            sg[nr-1]=new Segment2D(V[nr-1],V[0]);
            for (int i=1;i<nr-1;i++) U[i]=new Unghi2D(V[i],V[i-1],V[i+1]);
            U[0]=new Unghi2D(V[0],V[nr-1],V[1]);
            U[nr-1]=new Unghi2D(V[nr-1],V[nr-2],V[0]);
            tip="Triunghi";
      }
      public Triunghi2D(Triunghi2D T) {
            nr=T.nr;
            V=new Punct2D[nr];
            sg=new Segment2D[nr];
            U=new Unghi2D[nr];
            for (int i=0;i<nr;i++)
             {
               V[i]=new Punct2D(T.V[i]);
               sg[i]=new Segment2D(T.sg[i]);
               U[i]=new Unghi2D(T.U[i]);
             }
            tip="Triunghi";
      }
      //functii modificator
      public void setare(Punct2D P1, Punct2D P2, Punct2D P3) {
            V[0]=new Punct2D(P1);
            V[1]=new Punct2D(P2);
            V[2]=new Punct2D(P3);
            for (int i=0;i<nr-1;i++) sg[i]=new Segment2D(V[i],V[i+1]);
            sg[nr-1]=new Segment2D(V[nr-1],V[0]);
            for (int i=1;i<nr-1;i++) U[i]=new Unghi2D(V[i],V[i-1],V[i+1]);
            U[0]=new Unghi2D(V[0],V[nr-1],V[1]);
            U[nr-1]=new Unghi2D(V[nr-1],V[nr-2],V[0]);
      }
      public void setare(int caz, double a, double b, double c) {
            if (caz==1) {   //cazul LLL
               V[0]=new Punct2D(0,0);
               V[1]=new Punct2D(a,0);
               double u=(a*a+c*c-b*b)/(2*a*c);
               double u1=Math.acos(u);
               V[2]=new Punct2D(c*u, c*Math.sin(u1));
            }
            else if (caz==2) {   //cazul LUL
               V[0]=new Punct2D(0,0);
               V[1]=new Punct2D(a,0);
               double u=180-b;
            V[2]=new Punct2D(a+c*Math.cos(Math.toRadians(u)), c*Math.sin(Math.toRadians(u)));
            }
```

```
                else {     //cazul ULU
                   V[0]=new Punct2D(0,0);
                   V[1]=new Punct2D(b,0);
                   double lat=b*Math.sin(Math.toRadians(c))/Math.sin(Math.toRadians(180-a-c));
                 V[2]=new Punct2D(lat*Math.cos(Math.toRadians(a)),lat*Math.sin(Math.toRadians(a)));
                }
                for (int i=0;i<nr-1;i++) sg[i]=new Segment2D(V[i],V[i+1]);
                sg[nr-1]=new Segment2D(V[nr-1],V[0]);
                for (int i=1;i<nr-1;i++) U[i]=new Unghi2D(V[i],V[i-1],V[i+1]);
                U[0]=new Unghi2D(V[0],V[nr-1],V[1]);
                U[nr-1]=new Unghi2D(V[nr-1],V[nr-2],V[0]);
        }
        public void setare(Triunghi2D T) {
                super.setare(T);
        }
        //determina aria triunghiului
        public double arie() {
                double a,b,c,S,p;
                a=sg[0].getLungime();
                b=sg[1].getLungime();
                c=sg[2].getLungime();
                p=(a+b+c)/2;
                S=Math.sqrt(p*(p-a)*(p-b)*(p-c));
                return S;
        }
        //verifica daca doua triunghiuri sunt asemenea
        public boolean asemenea(Triunghi2D T) {
                boolean cod=false;
                double a1,b1,c1,a2,b2,c2;
                a1=sg[0].getLungime();
                b1=sg[1].getLungime();
                c1=sg[2].getLungime();
                a2=T.sg[0].getLungime();
                b2=T.sg[1].getLungime();
                c2=T.sg[2].getLungime();
                if (a1/a2==b1/b2 && a1/a2==c1/c2) cod=true;
                if (a1/a2==b1/c2 && a1/a2==c1/b2) cod=true;
                if (a1/b2==b1/a2 && a1/b2==c1/c2) cod=true;
                if (a1/b2==b1/c2 && a1/b2==c1/a2) cod=true;
                if (a1/c2==b1/a2 && a1/c2==c1/b2) cod=true;
                if (a1/c2==b1/b2 && a1/c2==c1/c2) cod=true;
                return cod;
        }
        //verifica daca doua triunghiuri coincid
        public boolean coincid(Triunghi2D t) {
                return V[0].coincid(t.V[0]) && V[1].coincid(t.V[1]) && V[2].coincid(t.V[2]);
        }
        //verifica daca triunghiul este isoscel
        public boolean isoscel() {
        return  sg[0].getLungime()==sg[1].getLungime()  ||  sg[0].getLungime()==sg[2].getLungime()    ||
sg[1].getLungime()==sg[2].getLungime();
        }
        //verifica daca triunghiul este echilateral
        public boolean echilateral() {
        return sg[0].getLungime()==sg[1].getLungime() && sg[0].getLungime()==sg[2].getLungime();
        }
        //verifica daca triunghiul este dreptunghic
```

```
	public boolean dreptunghic() {
		return U[0].getMasura()==90 || U[1].getMasura()==90 || U[2].getMasura()==90;
	}
	//verifica daca triunghiul este obtuzunghic
	public boolean obtuzunghic() {
		return U[0].getMasura()>90 || U[1].getMasura()>90 || U[2].getMasura()>90;
	}
	//verifica daca triunghiul este ascutitunghic
	public boolean ascutitunghic() {
		return !dreptunghic() && !obtuzunghic();
	}
	//determina mediatoarea unei laturi specificate a triunghiului
	public Dreapta2D mediatoare(Segment2D s) {
		return new Dreapta2D(s.mediatoare());
	}
	//determina piciorul bisectoarei interioare a unui unghi specificat al triunghiului
	public Punct2D picior_bisectoare(Punct2D p) {
		int i=0;
		if (p.coincid(V[0])) i=1;
		if (p.coincid(V[1])) i=2;
		if (p.coincid(V[2])) i=0;
		double x,y,k;
		if (i==1) {
			k=V[0].distanta(V[1])/V[0].distanta(V[2]);
			x=(V[1].getX()+k*V[2].getX())/(1+k);
			y=(V[1].getY()+k*V[2].getY())/(1+k);
		}
		else if (i==2) {
			k=V[1].distanta(V[2])/V[1].distanta(V[0]);
			x=(V[2].getX()+k*V[0].getX())/(1+k);
			y=(V[2].getY()+k*V[0].getY())/(1+k);
		}
		else {
			k=V[2].distanta(V[0])/V[2].distanta(V[1]);
			x=(V[0].getX()+k*V[1].getX())/(1+k);
			y=(V[0].getY()+k*V[1].getY())/(1+k);
		}
		return new Punct2D(x,y);
	}
	//determina bisectoarea interioara a unui unghi specificat al triunghiului
	public Semidreapta2D bisectoare_int(Punct2D p) {
		return new Semidreapta2D(p,picior_bisectoare(p));
	}
	//determina lungimea bisectoarei interioare dusa dintr-un varf dat al triunghiului
	public double lungime_bisectoare(Punct2D p) {
		double lung;
		if (p.coincid(V[0]))
		  lung=sg[2].lungime()*sg[0].lungime()*(1-(sg[1].lungime()/(sg[2].lungime()
	+sg[0].lungime()))*(sg[1].lungime()/(sg[2].lungime()+sg[0].lungime())));
		else if (p.coincid(V[1]))
			lung=sg[1].lungime()*sg[0].lungime()*(1-(sg[2].lungime()/(sg[1].lungime()
+sg[0].lungime()))*(sg[2].lungime()/(sg[1].lungime()+sg[0].lungime())));
		else		lung=sg[1].lungime()*sg[2].lungime()*(1-(sg[0].lungime()/(sg[1].lungime()
+sg[2].lungime()))*(sg[0].lungime()/(sg[1].lungime()+sg[2].lungime())));
		return lung;
	}
	//determina bisectoare exterioara a unui unghi specificat al triunghiului
```

```
public Semidreapta2D bisectoare_ext(Punct2D p) {
        Semidreapta2D bint=new Semidreapta2D(bisectoare_int(p));
        double u=bint.getUnghi();
        return new Semidreapta2D(bint.getOrigine(),u+90);
}
//determina mediana dusa dintr-un varf dat al triunghiului
public Segment2D mediana(Punct2D p) {
        int i=0;
        if (p.apartine(V[0])) i=1;
        if (p.apartine(V[1])) i=2;
        if (p.apartine(V[2])) i=0;
        Punct2D M=new Punct2D(sg[i].mijloc());
        return new Segment2D(p,M);
}
//determina linia mijlocie determinata de o latura a triunghiului
public Segment2D linie_mijlocie(Segment2D s) {
        Segment2D lm;
        if (s.coincid(sg[0])) lm=new Segment2D(sg[1].mijloc(),sg[2].mijloc());
        else if (s.coincid(sg[1])) lm=new Segment2D(sg[0].mijloc(),sg[2].mijloc());
        else  lm=new Segment2D(sg[1].mijloc(),sg[0].mijloc());
        return lm;
}
//determina piciorul simedianei dusa dintr-un varf dat al triunghiului
public Punct2D picior_simediana(Punct2D p) {
        int i=0;
        if (p.apartine(V[0])) i=1;
        if (p.apartine(V[1])) i=2;
        if (p.apartine(V[2])) i=0;
        Punct2D M=new Punct2D(sg[i].mijloc());
        double x,y,k;
        if (i==1) {
                k=V[0].distanta(V[1])/V[0].distanta(V[2]);
                x=(V[1].getX()+k*V[2].getX())/(1+k);
                y=(V[1].getY()+k*V[2].getY())/(1+k);
        }
        else if (i==2) {
                k=V[1].distanta(V[2])/V[1].distanta(V[0]);
                x=(V[2].getX()+k*V[0].getX())/(1+k);
                y=(V[2].getY()+k*V[0].getY())/(1+k);
        }
        else {
                k=V[2].distanta(V[0])/V[2].distanta(V[1]);
                x=(V[0].getX()+k*V[1].getX())/(1+k);
                y=(V[0].getY()+k*V[1].getY())/(1+k);
        }
        Punct2D D=new Punct2D(x,y);
        Simetrie2D s=new Simetrie2D(D);
        return new Punct2D(s.izometrie(M));
}
//determina simediana dusa dintr-un varf dat al triunghiului
public Segment2D simediana(Punct2D p) {
        return new Segment2D(p,picior_simediana(p));
}
//determina inaltimea care trece printr-un varf dat al triunghiului
public Dreapta2D inaltime(Punct2D p) {
        int i=0;
        if (p.coincid(V[0])) i=1;
```

```
                if (p.coincid(V[1])) i=2;
                if (p.coincid(V[2])) i=0;
                return new Dreapta2D(sg[i].getSuport().perpendiculara(p));
        }
        //determina lungimea inaltimii dusa dintr-un varf dat al triunghiului
        public double lungime_inaltime(Punct2D p) {
                double lung;
                if (p.coincid(V[0])) lung=2*arie()/sg[1].lungime();
                else if (p.coincid(V[1])) lung=2*arie()/sg[2].lungime();
                else  lung=2*arie()/sg[0].lungime();
                return lung;
        }
        //determina piciorul inaltimii care trece printr-un varf dat al triunghiului
        public Punct2D picior_inaltime(Punct2D p) {
                int i=0;
                if (p.coincid(V[0])) i=1;
                if (p.coincid(V[1])) i=2;
                if (p.coincid(V[2])) i=0;
                return new Punct2D(inaltime(p).intersectie(new Dreapta2D(sg[i].getSuport())));
        }
        //determina punctul de intersectie al mediatoarelor
        public Punct2D O() {
                Dreapta2D d1=new Dreapta2D(sg[0].mediatoare());
                Dreapta2D d2=new Dreapta2D(sg[1].mediatoare());
                Punct2D o=new Punct2D(d1.intersectie(d2));
                return o;
        }
        //determina punctul de intersectie al bisectoarelor interioare
        public Punct2D I() {
                Dreapta2D d1=new Dreapta2D(V[0],picior_bisectoare(V[0]));
                Dreapta2D d2=new Dreapta2D(V[1],picior_bisectoare(V[1]));
                Punct2D i=new Punct2D(d1.intersectie(d2));
                return i;
        }
        //determina punctul de intersectie al inaltimilor
        public Punct2D H() {
                return new Punct2D(inaltime(V[0]).intersectie(inaltime(V[1])));
        }
        //determina punctul de intersectie al medianelor
        public Punct2D G() {
                return new Punct2D(mediana(V[0]).getSuport().intersectie(mediana(V[1]).getSuport()));
        }
        //determina centrul cercului celor 9 puncte
        public Punct2D W() {
                Segment2D s=new Segment2D(O(),H());
                return s.mijloc();
        }
        //punctul lui Lemoine
        //determina punctul de intersectie al simedianelor
        public Punct2D K() {
    return new Punct2D(simediana(V[0]).getSuport().intersectie(simediana(V[1]).getSuport()));
        }
        //determina punctul lui Nagel
        public Punct2D pct_Nagel() {
                Cerc2D c1=new Cerc2D(cerc_exinscris(V[0]));
                Punct2D p1=new Punct2D(c1.punct_tangenta(sg[1].getSuport()));
                c1=new Cerc2D(cerc_exinscris(V[1]));
```

```
        Punct2D p2=new Punct2D(c1.punct_tangenta(sg[2].getSuport()));
        return new Punct2D((new Dreapta2D(V[0],p1)).intersectie(new Dreapta2D(V[1],p2)));
    }
    //determina punctul lui Gergonne
    public Punct2D pct_Gergonne() {
        Cerc2D c1=new Cerc2D(cerc_inscris());
        Punct2D p1=new Punct2D(c1.punct_tangenta(sg[1].getSuport()));
        Punct2D p2=new Punct2D(c1.punct_tangenta(sg[2].getSuport()));
        return new Punct2D((new Dreapta2D(V[0],p1)).intersectie(new Dreapta2D(V[1],p2)));
    }
    //determina punctul lui Vecten
    public Punct2D pct_Vecten() {
        Patrat2D p1=new Patrat2D(V[0],V[1]);
        Semiplan2D sp=new Semiplan2D(sg[0].getSuport(),V[2]);
        Punct2D C1=new Punct2D(p1.diagonala(V[0]).mijloc());
        if (sp.apartine(C1)) {
            Simetrie2D s=new Simetrie2D(sg[0].getSuport());
            C1.setare(s.izometrie(C1));
        }
        Patrat2D p2=new Patrat2D(V[1],V[2]);
        sp=new Semiplan2D(sg[1].getSuport(),V[0]);
        Punct2D A1=new Punct2D(p2.diagonala(V[1]).mijloc());
        if (sp.apartine(A1)) {
            Simetrie2D s=new Simetrie2D(sg[1].getSuport());
            A1.setare(s.izometrie(A1));
        }
        return new Punct2D((new Dreapta2D(V[0],A1)).intersectie(new Dreapta2D(V[2],C1)));
    }
    //determina dreapta lui Euler
    public Dreapta2D dr_Euler() {
        return new Dreapta2D(O(),H());
    }
    //determina dreapta lui Simson
    public Dreapta2D dr_Simson(Punct2D P) {
        Dreapta2D d1=new Dreapta2D(sg[0].getSuport().perpendiculara(P));
        Punct2D P1=new Punct2D(d1.intersectie(sg[0].getSuport()));
        d1=new Dreapta2D(sg[1].getSuport().perpendiculara(P));
        Punct2D P2=new Punct2D(d1.intersectie(sg[1].getSuport()));
        return new Dreapta2D(P1,P2);
    }
    //determina dreapta ortica
    public Dreapta2D dr_ortica() {
        Punct2D A3=new Punct2D(picior_inaltime(V[0]));
        Punct2D B3=new Punct2D(picior_inaltime(V[1]));
        Punct2D C3=new Punct2D(picior_inaltime(V[2]));
    Punct2D A1=new Punct2D((new Dreapta2D(V[1],V[2])).intersectie(new Dreapta2D(B3,C3)));
    Punct2D B1=new Punct2D((new Dreapta2D(V[0],V[2])).intersectie(new Dreapta2D(A3,C3)));
        return new Dreapta2D(A1,B1);
    }
    //determina dreapta lui Lemoine
    public Dreapta2D dr_Lemoine() {
        Cerc2D C=new Cerc2D(cerc_circumscris());
        Dreapta2D tgA=new Dreapta2D((new Dreapta2D(O(),V[0])).perpendiculara(V[0]));
        Punct2D A1=new Punct2D(tgA.intersectie(new Dreapta2D(V[1],V[2])));
        Dreapta2D tgB=new Dreapta2D((new Dreapta2D(O(),V[1])).perpendiculara(V[1]));
        Punct2D B1=new Punct2D(tgB.intersectie(new Dreapta2D(V[0],V[2])));
        return new Dreapta2D(A1,B1);
```

```
}
//triunghiul ortic
public Triunghi2D triunghi_ortic() {
return new Triunghi2D(picior_inaltime(V[0]),picior_inaltime(V[1]),picior_inaltime(V[2]));
}
//triunghiul median
public Triunghi2D triunghi_median() {
        Punct2D p1=new Punct2D(sg[0].mijloc());
        Punct2D p2=new Punct2D(sg[1].mijloc());
        Punct2D p3=new Punct2D(sg[2].mijloc());
        return new Triunghi2D(p1,p2,p3);
}
//determina cercul circumscris triunghiului
public Cerc2D cerc_circumscris() {
        double r=sg[0].getLungime()*sg[1].getLungime()*sg[2].getLungime();
        r=r/(4*arie());
        return new Cerc2D(O(),r);
}
//determina cercul inscris in triunghi
public Cerc2D cerc_inscris() {
        double p=(sg[0].getLungime()+sg[1].getLungime()+sg[2].getLungime())/2;
        double r=arie()/p;
        return new Cerc2D(I(),r);
}
//determina cercul celor 9 puncte
public Cerc2D cerc_Euler() {
        double r=sg[0].getLungime()*sg[1].getLungime()*sg[2].getLungime();
        r=r/(8*arie());
        return new Cerc2D(W(),r);
}
//determina primul cerc al lui Lemoine
public Cerc2D cerc_Lemoine() {
        double r=sg[0].getLungime()*sg[1].getLungime()*sg[2].getLungime();
        r=r/(8*arie());
        return new Cerc2D(K(),r);
}
//determina un cerc exinscris triunghiului
public Cerc2D cerc_exinscris(Punct2D p) {
        int i;
        if (p.apartine(V[0])) i=0;
        else if (p.apartine(V[1])) i=1;
        else i=2;
        Dreapta2D d1,d2,d3;
        if (i==0) {
                d1=new Dreapta2D(bisectoare_int(V[0]).getSuport());
                d2=new Dreapta2D(bisectoare_int(V[1]).getSuport());
                d3=new Dreapta2D(d2.perpendiculara(V[1]));
        }
        else if (i==1) {
                d1=new Dreapta2D(bisectoare_int(V[1]).getSuport());
                d2=new Dreapta2D(bisectoare_int(V[0]).getSuport());
                d3=new Dreapta2D(d2.perpendiculara(V[0]));
        }
        else {
                d1=new Dreapta2D(bisectoare_int(V[2]).getSuport());
                d2=new Dreapta2D(bisectoare_int(V[1]).getSuport());
                d3=new Dreapta2D(d2.perpendiculara(V[1]));
```

```
                }
                Punct2D I=new Punct2D(d3.intersectie(d1));
                if (i==0) d1.setare(V[0],V[1]);
                else if (i==1) d1.setare(V[1],V[0]);
                else d1.setare(V[2],V[1]);
                d2.setare(d1.perpendiculara(I));
                Punct2D M=new Punct2D(d2.intersectie(d1));
                double r=I.distanta(M);
                return new Cerc2D(I,r);
        }
        //determina cercul lui Taylor
        public Cerc2D cerc_Taylor() {
                Punct2D A1=new Punct2D(picior_inaltime(V[0]));
                Dreapta2D d1=new Dreapta2D((sg[0].getSuport()).perpendiculara(A1));
                Punct2D A2=new Punct2D(d1.intersectie(sg[0].getSuport()));
           d1=new Dreapta2D((sg[2].getSuport()).perpendiculara(A1));
                Punct2D A3=new Punct2D(d1.intersectie(sg[2].getSuport()));
                Punct2D B1=new Punct2D(picior_inaltime(V[1]));
                d1=new Dreapta2D((sg[0].getSuport()).perpendiculara(B1));
                Punct2D B2=new Punct2D(d1.intersectie(sg[0].getSuport()));
                return new Cerc2D(A2,A3,B2);
        }
}
```

Clasa **Elipsoid**

```
public class Elipsoid extends Element3D {
        protected Punct3D C;
        protected double a,b,c;
        //metode
          //constructori
        public Elipsoid() {
           C=new Punct3D(0,0,0);
           a=1;b=2;c=3;
           tip="Elipsoid";
           calcul_puncte();
        }
        public Elipsoid(double a, double b,double c) {
                C=new Punct3D(0,0,0);
           this.a=a;this.b=b;this.c=c;
           tip="Elipsoid";
           calcul_puncte();
        }
        public Elipsoid(Punct3D P,double a, double b,double c) {
                C=new Punct3D(P);
           this.a=a;this.b=b;this.c=c;
           tip="Elipsoid";
           calcul_puncte();
        }
        public Elipsoid(Elipsoid E) {
                C=new Punct3D(E.C);
           this.a=E.a;this.b=E.b;this.c=E.c;
           tip="Elipsoid";
           culoare=E.culoare;
                stil=E.stil;
                calcul_puncte();
```

```
        }
        //functii de acces
        public Punct3D getCentru() {
                return C;
        }
        public double getA() {
                return a;
        }
        public double getB() {
                return b;
        }
        public double getC() {
                return c;
        }
        //functii modificator
        public void setare(double a, double b,double c) {
                C=new Punct3D(0,0,0);
           this.a=a;this.b=b;this.c=c;
           tip="Elipsoid";
           calcul_puncte();
        }
        public void setare(Punct3D P,double a, double b,double c) {
                C=new Punct3D(P);
           this.a=a;this.b=b;this.c=c;
           tip="Elipsoid";
           calcul_puncte();
        }
        public void setare(Elipsoid E) {
                C=new Punct3D(E.C);
           this.a=E.a;this.b=E.b;this.c=E.c;
           tip="Elipsoid";
           culoare=E.culoare;
                stil=E.stil;
                calcul_puncte();
        }
          //volumul elipsoidului
        public double volum() {
                return 4*a*b*c*Math.PI/3;
        }
        //normala la elipsoid intr-un punct
        public Dreapta3D normala(Punct3D P, double u, double v ) {
                double p,q,r;
p=-b*c*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(v));
q=-a*c*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(u))*Math.sin(Math.toRadians(v));;
                r=-a*b*Math.sin(Math.toRadians(u))*Math.cos(Math.toRadians(u));
                return new Dreapta3D(P,p,q,r);
        }
        //planul tangent la elipsoid intr-un punct
        public Plan3D plan_tangent(Punct3D P, double u, double v ) {
                double A,B,C1,D;
A=-b*c*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(v));
B=-a*c*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(u))*Math.sin(Math.toRadians(v));;
                C1=-a*b*Math.sin(Math.toRadians(u))*Math.cos(Math.toRadians(u));
                D=-A*P.getX()-B*P.getY()-C1*P.getZ();
                return new Plan3D(A,B,C1,D);
        }
        //metoda care verifica daca un punct apartine obiectului geometric
```

```
	public boolean apartine(Punct3D P){
		return   ((P.getX()-C.getX())*(P.getX()-C.getX()))/(a*a)+((P.getY()-C.getY())*(P.getY()-
C.getY()))/(b*b)+((P.getZ()-C.getZ())*(P.getZ()-C.getZ()))/(c*c)==1;
	}
	// metoda care returneaza un punct apartinand elipsoidului cand se specifica unghiurile
	public Punct3D punct(double u, double v) {
		double x,y,z;
		x=C.getX()+a*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(v));
		y=C.getY()+b*Math.cos(Math.toRadians(u))*Math.sin(Math.toRadians(v));
		z=C.getZ()+c*Math.sin(Math.toRadians(u));
		return new Punct3D(x,y,z);
	}
	//metoda care determina punctele de pe suprafata
	public Punct3D[][] calcul_puncte() {
		Punct3D[][] puncte=new Punct3D[31][61];
		double x,y,z,u,v;
		int i,j;
		for (i=0,u=-90;u<=90;i++,u+=6)
		   for (j=0,v=0;v<=360;j++,v+=6) {
			x=C.getX()+a*Math.cos(Math.toRadians(u))*Math.cos(Math.toRadians(v));
			y=C.getY()+b*Math.cos(Math.toRadians(u))*Math.sin(Math.toRadians(v));
			z=C.getZ()+c*Math.sin(Math.toRadians(u));
			puncte[i][j]=new Punct3D(x,y,z);
		   }
		return puncte;
	}
	//metoda care modifica punctele de pe suprafata printr-o izometrie
	public void setare_puncte(Punct3D[][] p) {
		int i,j;
		double u,v;
		for (i=0,u=-90;u<=90;i++,u+=6)
		   for (j=0,v=0;v<=360;j++,v+=6)
		      puncte[i][j].setare(p[i][j].getX(),p[i][j].getY(),p[i][j].getZ());
	}
	//metoda care determina proiectiile punctelor de pe suprafata pe planul de proiectie
	public Punct2D[][] matrice(int latime, int inaltime, Proiectie pr) {
		Punct2D pct[][]=new Punct2D[31][61];
		double x,y,z;
		double u,v,xx,yy,xe,ye;
		int i,j;
		for (i=0,u=-90;u<=90;i++,u+=6)
		   for (j=0,v=0;v<=360;j++,v+=6) {
		      xx=pr.getX(puncte[i][j]);
		      yy=pr.getY(puncte[i][j]);
		      xe=Desen3D.xecran(latime,(int)(latime/2),xx);
		      ye=Desen3D.yecran(inaltime,(int)(inaltime/2),yy);
		      pct[i][j]=new Punct2D(xe,ye);
		   }
		return pct;
	}
	//metoda de desenare a elementului geometric
	public void desenare(Graphics2D g, int latime, int inaltime, Proiectie pr) {
		figura=new GeneralPath();
		Punct2D pct[][]=new Punct2D[31][61];
		pct=matrice(latime,inaltime,pr);
		int i,j;
		for (i=0;i<30;i++)
```

```
                    for (j=0;j<60;j++) {
                        GeneralPath fig=new GeneralPath();
                        fig.moveTo((float)(pct[i][j].getX()),(float)(pct[i][j].getY()));
                        fig.lineTo((float)(pct[i+1][j].getX()),(float)(pct[i+1][j].getY()));
                        fig.lineTo((float)(pct[i+1][j+1].getX()),(float)(pct[i+1][j+1].getY()));
                        fig.lineTo((float)(pct[i][j+1].getX()),(float)(pct[i][j+1].getY()));
                        fig.closePath();
                        if (culoare.equals(Color.black)) g.setColor(new Color(0,0,0,100));
                        if (culoare.equals(Color.gray))
g.setColor(new Color((Color.gray).getRed(),(Color.gray).getGreen(),(Color.gray).getBlue(),100));
                        if (culoare.equals(new Color(155,65,30)))
                           g.setColor(new Color(155,65,30,100));
                        if (culoare.equals(Color.green)) g.setColor(new Color(0,255,0,80));
                        if (culoare.equals(Color.blue)) g.setColor(new Color(0,0,250,100));
                        if (culoare.equals(new Color(155,55,150)))
                           g.setColor(new Color(155,55,150,120));
                        if (culoare.equals(Color.red)) g.setColor(new Color(255,0,0,110));
                        if (culoare.equals(new Color(250,70,150)))
                           g.setColor(new Color(250,70,150,120));
                        if (culoare.equals(Color.yellow)) g.setColor(new Color(255,0,0,70));
                        if (culoare.equals(Color.white))
g.setColor(new Color((Color.gray).getRed(),(Color.gray).getGreen(),(Color.gray).getBlue(),100));
                        g.fill(fig);
                    }
                for (i=0;i<=30;i++) {
                        figura.moveTo((float)(pct[i][0].getX()),(float)(pct[i][0].getY()));
                        for (j=1;j<=60;j++)
                            figura.lineTo((float)(pct[i][j].getX()),(float)(pct[i][j].getY()));
                }
                for (j=0;j<=60;j++) {
                   figura.moveTo((float)(pct[0][j].getX()),(float)(pct[0][j].getY()));
                   for (i=1;i<=30;i++) figura.lineTo((float)(pct[i][j].getX()),(float)(pct[i][j].getY()));
                }
                g.setColor(culoare);
                g.setStroke(stil);
                g.draw(figura);
        }
}
```