Visual interactive environment for doing geometrical constructions

ANCA IORDAN[1], GEORGE SAVII[2], MANUELA PĂNOIU[1], CAIUS PĂNOIU[1]
[1] Technical University of Timişoara, Engineering Faculty of Hunedoara,
Revolutiei 5, 331128 Hunedoara
[2] Technical University of Timişoara, Mechanical Engineering Faculty,
Mihai Viteazu 1, 300222 Timişoara
ROMANIA
anca.iordan@fih.upt.ro

Abstract: - In this work will be presented the elaboration of an educational informatics system for doing geometry on a computer. In a way it replaces pencil, paper, ruler and compass with equivalent computer tools. The achieved informatics system will be able to be used for teachning Euclidean geometry, both in pre-university and in university education.

Key-Words: - Dynamical software, Euclidean geometry, Java, distance education.

1 Introduction

"Dynamic geometry software" has become a generic term for a class of geometry software environments where geometric objects can be continuously transformed on the screen by dragging, with only those features based on geometric properties remaining invariant. Accurate measurements can be performed, and the software incorporates Euclidean geometry tools, such as *angle bisector* and *perpendicular line*.

Laborde [1] asserts that dynamic drawings offer stronger visual evidence than a single static drawing: "A spatial property may emerge as an invariant in the movement whereas this might not be noticeable in one static drawing".

Love [2], on the other hand, questions the impact of readily produced computer images on the learner's ability to generate his/her own mental images, noting that "it is easy to become seduced by the visualisations to the extent of thinking that consideration of them is the purpose of using them in geometry".

Despite the strong feeling that the dynamic imagery associated with use of the software has the potential to play a significant part in geometric reasoning, concern has been expressed that dynamic geometry software is contributing to an empirical approach to school geometry [3], [4]. Instead, traditional geometry exercises have been adapted for the computer, and, of greater concern, geometry is being reduced to pattern-spotting in data generated by dragging and measurement of screen drawings, with little or no emphasis on theoretical geometry: "school mathematics is poised to incorporate powerful dynamic geometry tools in order merely to spot patterns and generate cases"[5].

Although it appears that there are many instances of dynamic geometry software being used merely to collect empirical data, it is also possible to use the software in ways that encourage geometric reasoning [6], [7]. The construction of geometric shapes which retain their properties and relationships when dragged, focuses students' attention on the relationships between properties. Other activities may require students to explore, make conjectures, and prove properties for a given geometric figure, or to model and investigate a dynamic physical situation in order to understand the effect of changing parameters [8].

Different modes of dragging of dynamic geometry figures may be used depending on the information which the user hopes to gain. If, for example, a student has constructed an isosceles triangle, dragging may be used as a check that the triangle will remain isosceles, confirming that the triangle has been appropriately constructed [9], [10], [11]. Dragging may also be used in exploratory tasks, where a figure is dragged in order to satisfy a particular visual constraint. This mode of dragging is often used in association with tracing the path of a point. Laborde and Laborde [12] suggest that this exploration provides a starting point for the conjecture that the path of point *A* is a circle, that in turn may lead students to the construction of a circle with the midpoint of *BC* as centre. Dragging may then be used as a check to confirm the conjecture. Arzarello [13] assert that students' use of dragging when investigating a problem in a dynamic geometry environment changes as they develop a greater understanding of the problem, and that the different modes of dragging play a part in the progression to deductive reasoning.

2 Presentation of the proposed informatics systems

In this paragraph will be presented the elaboration of an educational informatics system for doing geometry on a computer. In a way it replaces pencil, paper, ruler and compass with equivalent computer tools. With the mouse you "draw" in a window on screen, i.e. you place points, connect them by lines, erect perpendiculars, etc.

This functionality is not very exciting, but already useful if you want to do exact constructions. You can be sure that you exactly – only restricted by the floating point accuracy of the computer – hit an intersection point of two lines, or draw an exact parallel to some other line. Also some constructions that are tedious to do by hand are easily done with the computer, for example inversions at a circle (or conic). The additional possibility of rescaling a figure can help you if the construction you are doing exceeds the limits of the window.

For elaboration of the informatics educational system shall be used the Java programming language [14].

2.1 System's analysis

The informatics system will be described in a clear and concise manner by presenting the use cases, using the UML (unified modeling language) [15]. Each use case is a sequence of related transactions performed by an actor and the system in a dialogue. Representation of the uses cases' diagram is shown in figure 1.

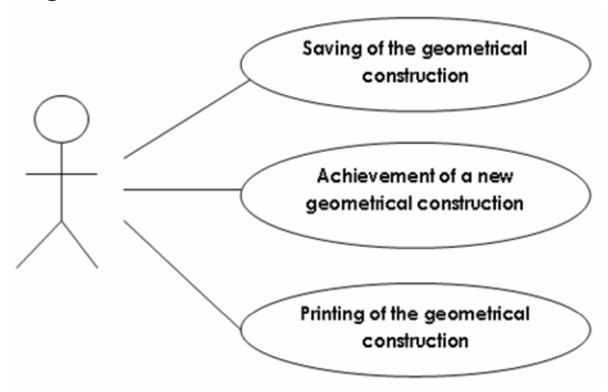
Saving of the geometrical construction
Achievement of a new geometrical construction
Printing of the geometrical construction

Fig. 1 The use cases' diagram

Activity diagrams [16] are used:

- To capture actions to be performed when an operation is executing (most common purpose);
- To capture the internal work in an objects;
- To show how a set of related actions may be performed;
- To show how an instance of s use-case may be performed;
- To show a business works in terms of actors, workflows, organization, and objects.

For each use case presented in the previous diagram we'll build activitiy diagram. Each diagram will specify the processes or algorithms which are behind the analysed use case. Figure 2 will present these diagrams.

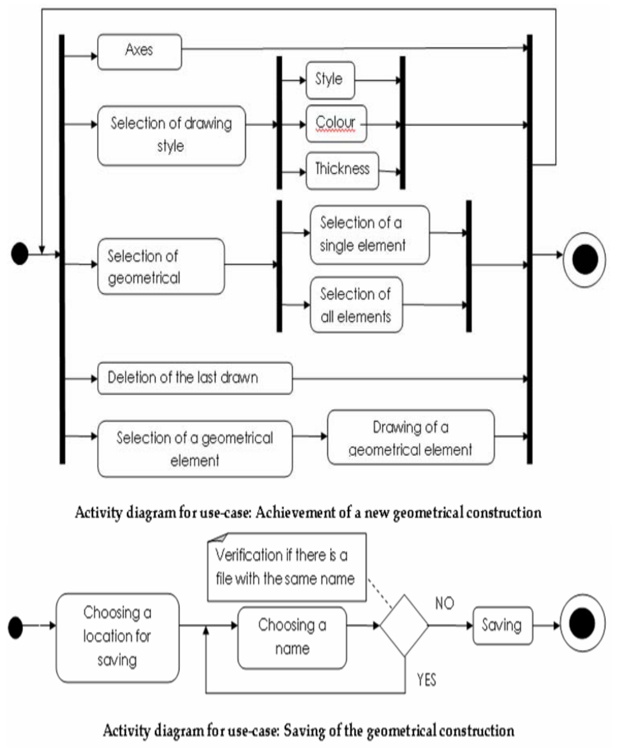
Axes
Style
Selection of drawing style
Colour
Thickness
Selection of a single element
Selection of geometrical
Selection of all elements
Deletion of the last drawn
Selection of a geometrical element
Drawing of a geometrical element
Activity diagram for use-case: Achievement of a new geometrical construction
Verification if there is a file with the same name
NO
Choosing a location for saving
Choosing a name
Saving
YES
Activity diagram for use-case: Saving of the geometrical construction

Fig. 2 Activity diagrams

2.2 System's designing

The conceptual modeling allows the identification of the most important concepts for the informatics system [17], [18]. Because classes means concepts, will be used the following classes to identify the plane geometry elements: point, line, semi-line, segment, semi-plane, conic, ellipse, circle, hyperbola, parabola, affine transformation, symmetry, rotation, translation, inversion.

The existing inheritance relationships between the classes previously presented can be represented by means of relationship diagrams between classes (Fig. 3).

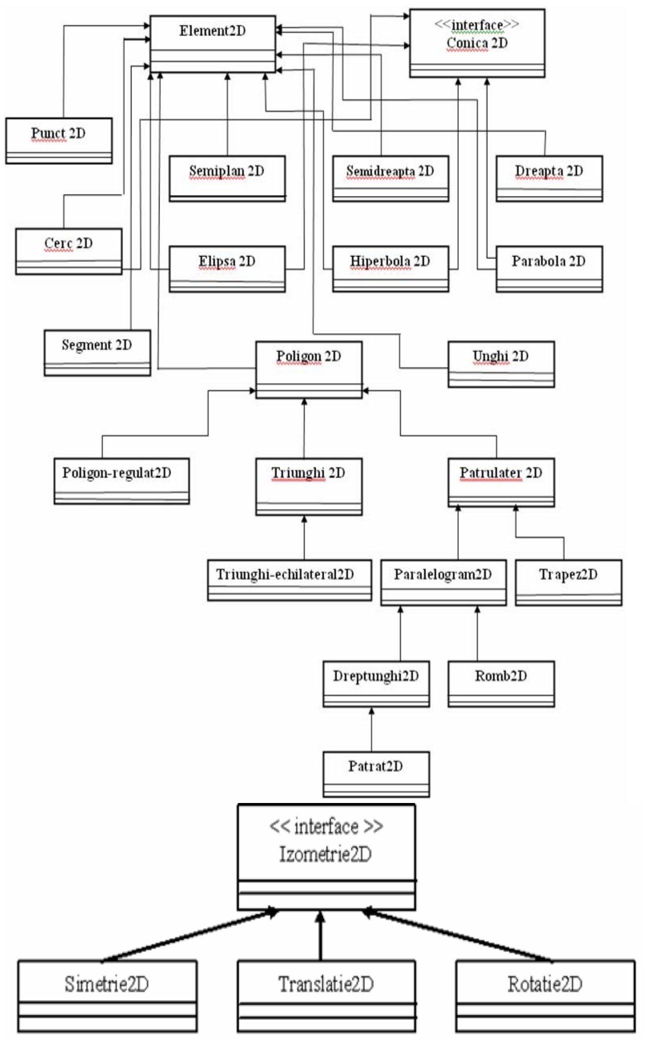

Fig. 3 Inheritance relationships

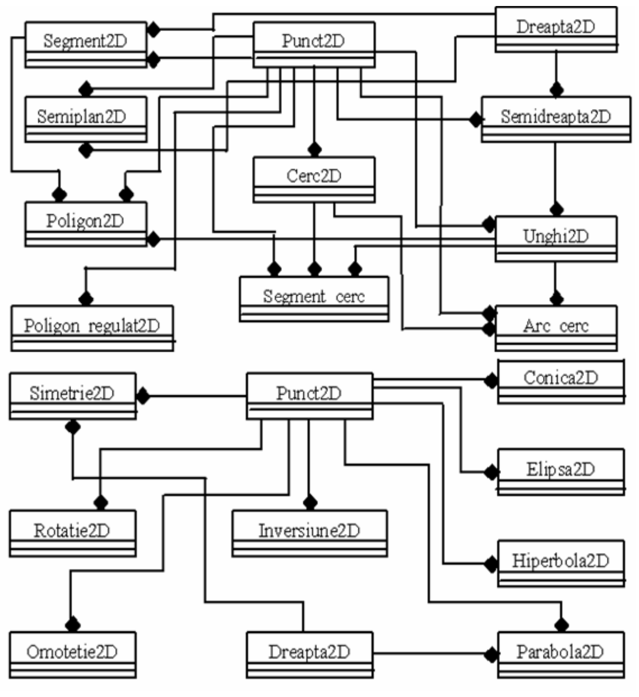

Fig. 4 Composition relationships

Between the classes' instances from the previously presented architecture there are mainly composition relationships. Further, we will present these relationships by means of the relationship diagrams between classes' instances (Fig. 4).

The presented classes' instances can be grouped by related properities and methods. The grouping is achieved by means of package. In figure 5 are explicitely presented the structures of the package which group objects from plane geometry. Between classes and classes' instances from a package there are inheritance and composition relationships, which where presented by means of the previous diagrams.

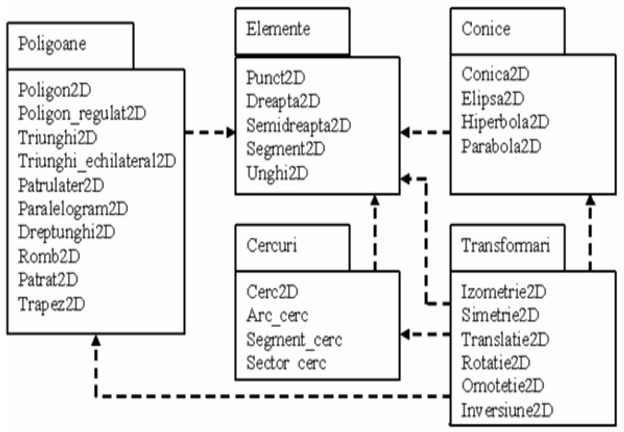

Fig. 5 Package diagram

The other two kinds of dynamic diagram fall into a category called Interaction diagrams. They both describe the flow of messages between objects. However, sequence diagrams focus on the order in which the messages are sent. They are very useful for describing the procedural flow through many objects. They are also quite useful for finding race conditions in concurrent systems.

The sequence diagram [19] is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them. However, an organization's business staff can find sequence diagrams useful to communicate how the business currently works by showing how various business objects interact. Besides documenting an organization's current affairs, a business-level sequence diagram can be used as a requirements document to communicate requirements for a future system implementation. During the requirements phase of a project, analysts can take use cases to the next level by providing a more formal level of refinement. When that occurs, use cases are often refined into one or more sequence diagrams. One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams.

The sequence diagrams for this software, illustrates in figure 6, are made with ArgoUML-0.24.

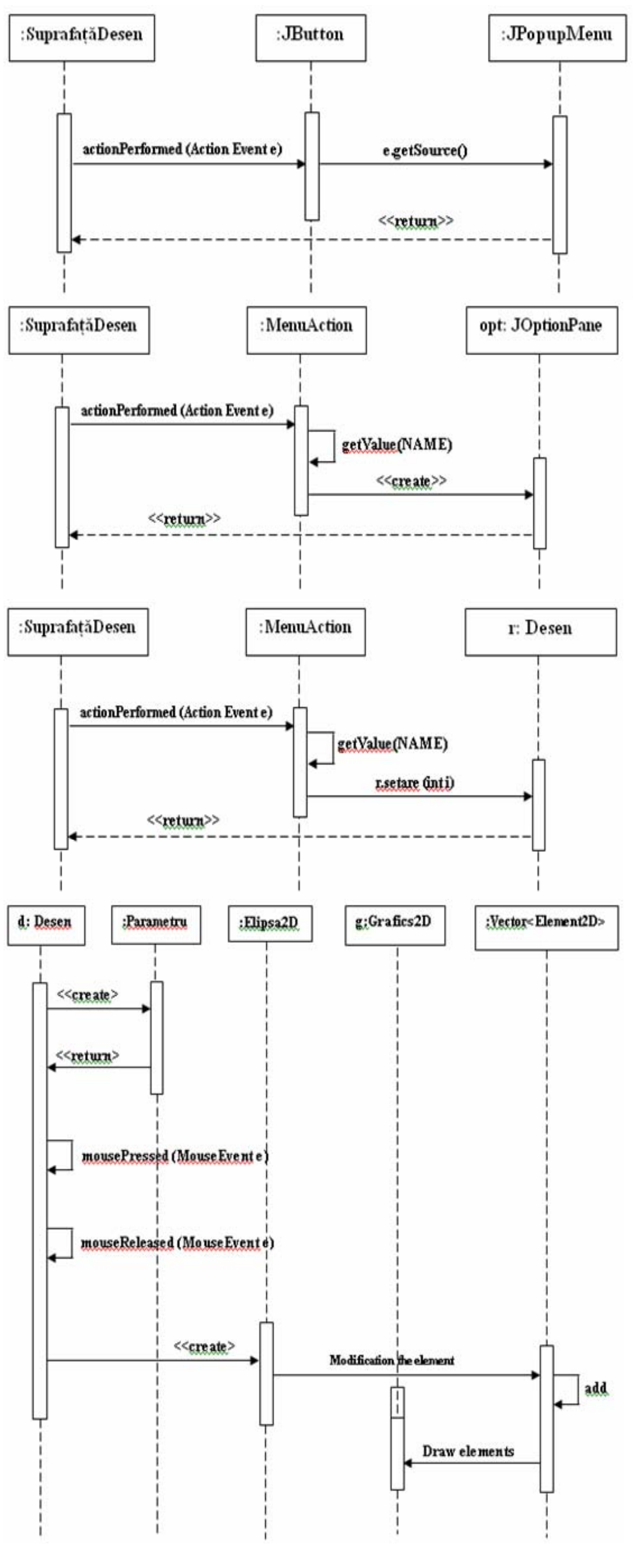

Fig. 6 Sequence diagrams

Collaboration diagram [20], on the other hand, focus upon the relationships between the objects. They are very useful for visualizing the way several objects collaborate to get a job done and for comparing a dynamic model with a static model. Collaboration and sequence diagrams describe the same information, and can be transformed into one another without difficulty. The choice between the two depends upon what the designer wants to make visually apparent.

In figure 7, 8, 9, 10 and 11 are illustrates collaboration diagrams.

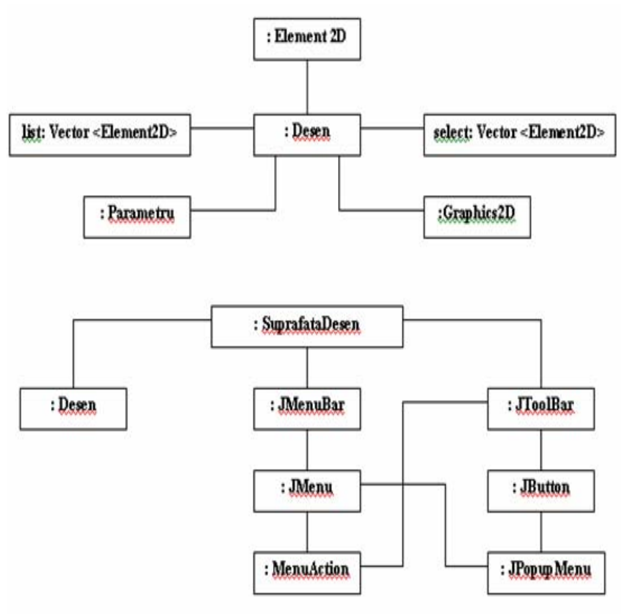

Fig. 7 Collaboration diagrams

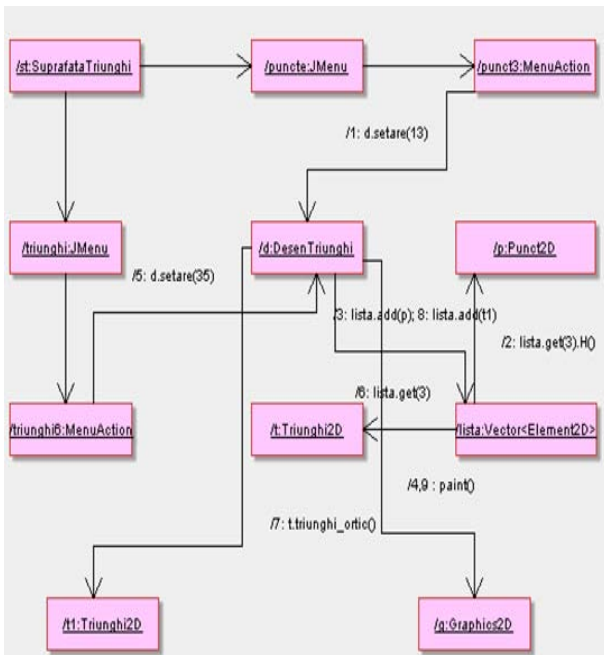

Fig. 8 Collaboration diagram for drawing the orthocenter and the orthic triangle

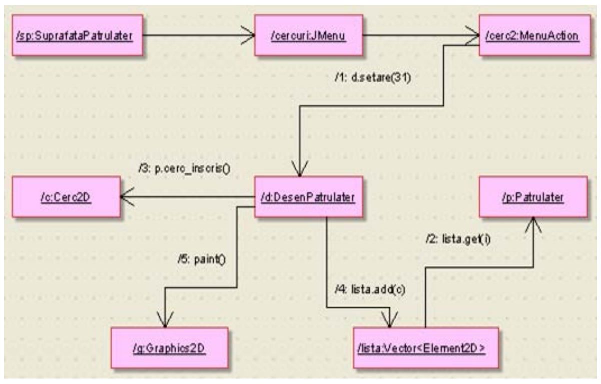

Fig. 9 Collaboration diagram for drawing the incircle of a quadrilateral

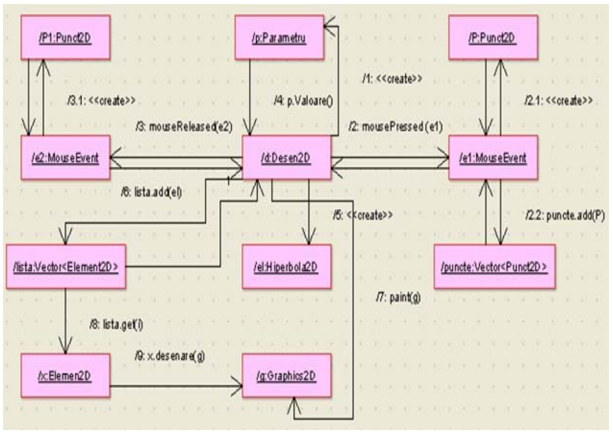

Fig. 10 Collaboration diagram for drawing a hyperbola

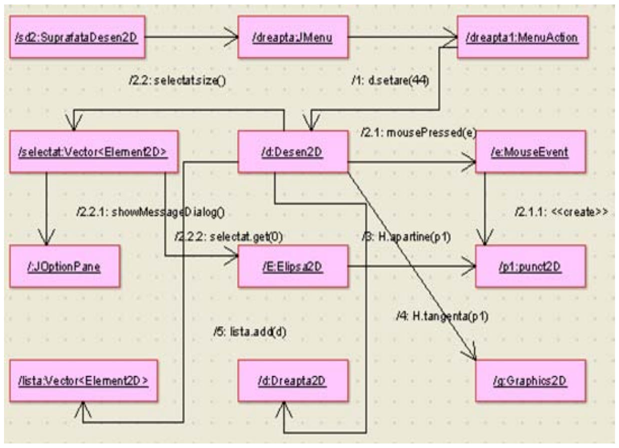

Fig. 11 Collaboration diagram for drawing the tangent and the normal to an ellipse

2.3 Component diagram

A component represents a modular, replaceable piece in the system [21]. Of primary importance are two well-defined interfaces: The required interface specifies formally which functionality the component expects from its environment. The provided interface specifies the functionality the component is able to provide (to other components).

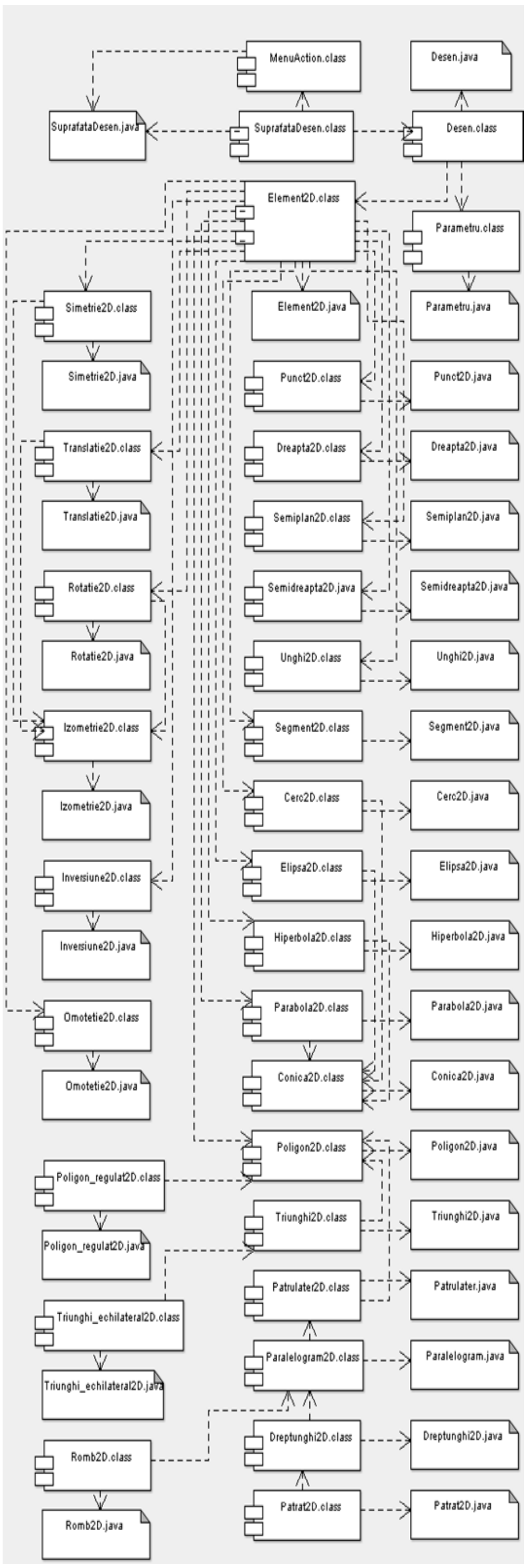

Fig. 12 Component diagram

Ideally, the interfaces capture the required and provided functionality in such a detailed manner that allows to exchange two components conforming to the same interfaces freely. As a consequence, we could substitute one component by another component without changing anything else of the rest of the system (component is replaceable).

In principle, all parts of the system that are modeled by a component can also be modeled by a class. It is possible also for a class to express which interface it provides and which it requires.

In figure 12 is presented the component diagram wich is made with ArgoUML-0.24.

2.4 User interface

The program's interface includes a menu bar, a bar with buttons corresponding to the most important operations and the drawing surface (Fig. 13).

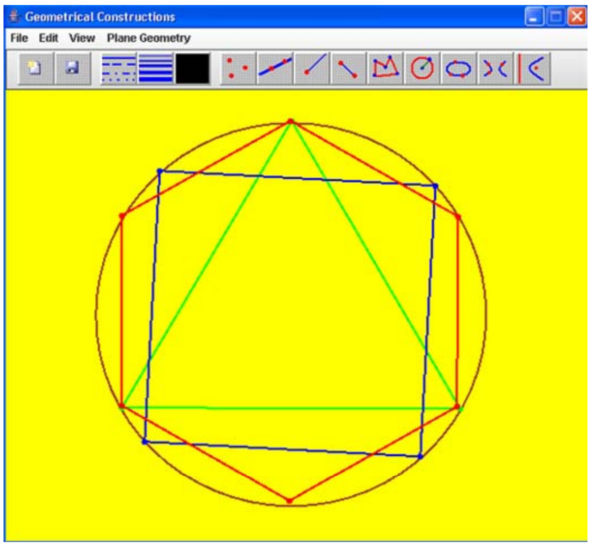

Fig. 13 An example for a construction that is easy with this software

Among the most important operations we mention :

- drawing-up of free points or points with certain properties, i.e. the middle of a segment, Newton's Point, Miquel's Point, Mathot's Point;
- drawing-up of certain lines or lines which fulfill certain conditions, i.e. the paralell to a line or the perpendicular to a line (Fig. 14), Euler line, Lemoine line, Newton line, Gauss line, Aubert line, bimedians for a given quadrilateral;
- drawing-up of triangles which fulfill certain conditions, i.e. orthic triangle, medial triangle;
- drawing-up of circles when is specified the centre and radius or when are selected three points which will identify the circle or circles which fulfill certain conditions, i.e. the incircle and the three excircles for a given triangle, Taylor's circle;
- selection of the geometrical transformations in plane: symmetry, rotation, translation, inversion, homothety, i.e. rotation of a parallelogram, rotation of a parabola, symmetry of a decagon, symmetry of an ellipse, translation of an octagon, translation of a hyperbola, homothety of a pentagon, homothety of an ellipse, inversion of a circle;
- possibility to move the geometrical construction;
- possibility to undertake parts of a geometrical construction and keeping them in the button's bar for a possible further reutilisation.

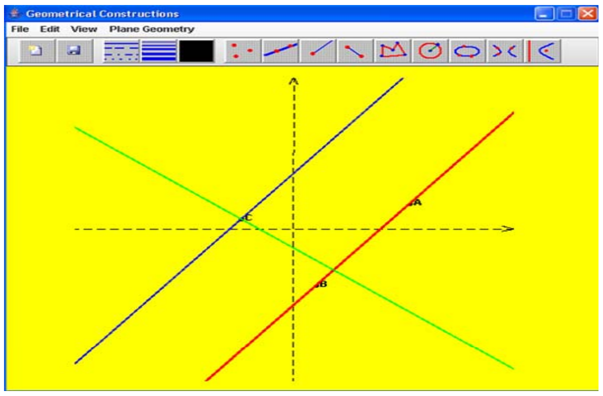

Fig. 14 Parallel and perpendicular to a line

The Lemoine line is draw by the following method of the class Triunghi2D (Fig. 15):

```
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); }
```

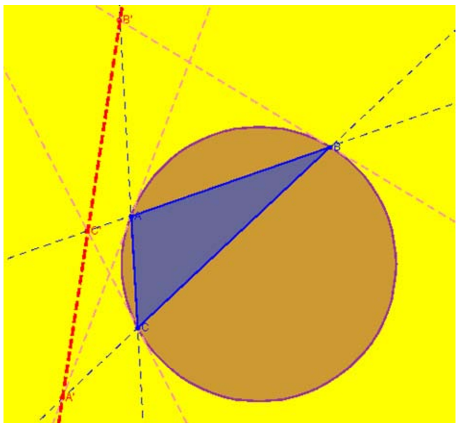

Fig. 15 Lemoine Line

The Medial triangle is draw by the following method of the class Triunghi2D (Fig. 16):

```
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); }
```

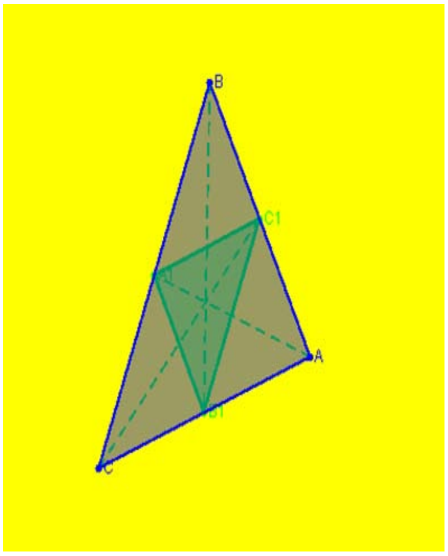

Fig. 16 Medial triangle

The method of class Triunghi2D which obtains the incircle (Fig. 17) is presented forwards:

```
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); }
```

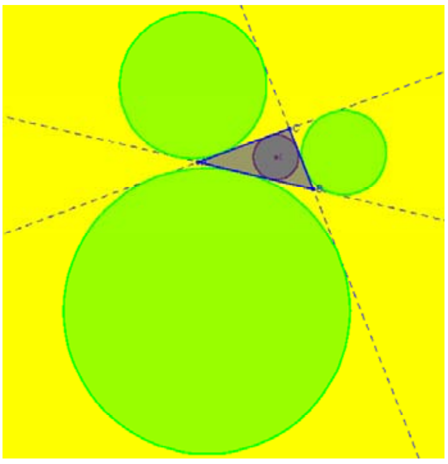
Fig. 17 The incircle and the three excircles for a given triangle

The implementation of the method which obtains Taylor's circle (Fig. 18) can be as follows:

```
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); }
```

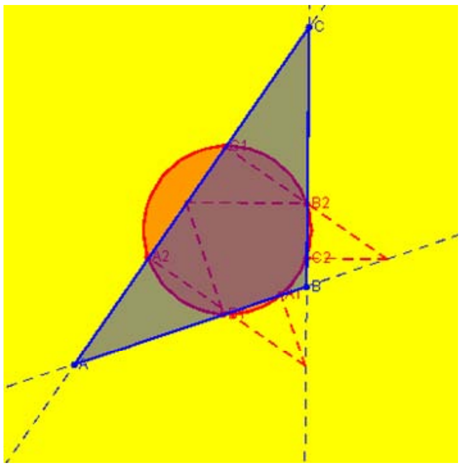

Fig. 18 Taylor's circle

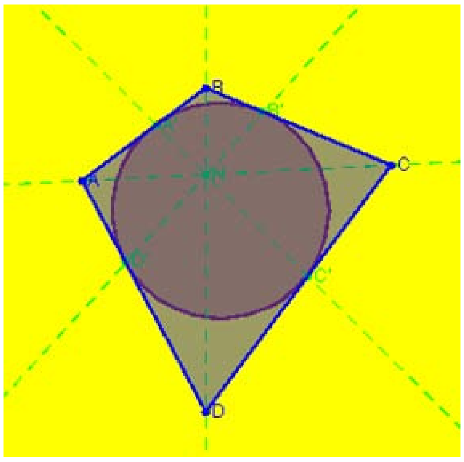

Fig. 19 Newton's Point

The Newton's Point is draw by the following method of the class Patrulater2D (Fig. 19):

```
public Punct2D pct_Newton() {
return new Punct2D(new Dreapta2D(
V[0],V[2]).intersectie(new Dreapta2D(V[1],V[3])));
}
```

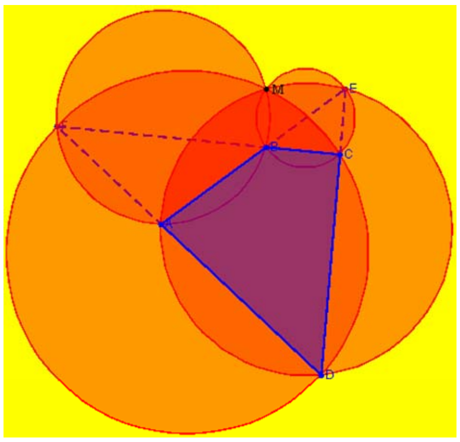

Fig. 20 Miquel's Point

The Miquel's Point is draw by the following method of the class Patrulater2D (Fig. 20):

```
public Punct2D pct_Miquel() {
Triunghi2D t1=new Triunghi2D(V[0],V[3],
diagonala3().getExtremitate1());
Triunghi2D t2=new Triunghi2D(V[0],V[1],
diagonala3().getExtremitate2());
Triunghi2D t3=new Triunghi2D(V[3],V[2],
diagonala3().getExtremitate2());
Cerc2D c1=new Cerc2D(t1.cerc_circumscris());
Cerc2D c2=new Cerc2D(t2.cerc_circumscris());
Cerc2D c3=new Cerc2D(t3.cerc_circumscris());
Punct2D M;
if (c1.tangente(c2)) M=new Punct2D(c1.
punct_tangenta(c2));
else if (c1.tangente(c3)) M=new Punct2D(c1.
punct_tangenta(c3));
else if (c3.tangente(c2)) M=new Punct2D(c3.
punct_tangenta(c2));
else {
        Punct2D p1=new Punct2D();
        Punct2D p2=new Punct2D();
        Punct2D p3=new Punct2D();
        Punct2D p4=new Punct2D();
        c1.puncte_secante(c2,p1,p2);
        c1.puncte_secante(c3,p3,p4);
        if (p1.coincid(p3)||p1.coincid(p4)) M=new
Punct2D(p1);
        else M=new Punct2D(p2);
    }
return M; }
```

The method of class Patrulater2D which obtains the Mathot's Point (Fig. 21) is presented forwards:

```
public Punct2D pct_Mathot() {
Dreapta2D d1=new Dreapta2D((sg[2].getSuport()).
perpendiculara(sg[0].mijloc()));
Dreapta2D d2=new Dreapta2D((sg[3].getSuport()).
perpendiculara(sg[1].mijloc()));
return new Punct2D(d1.intersectie(d2)); }
```

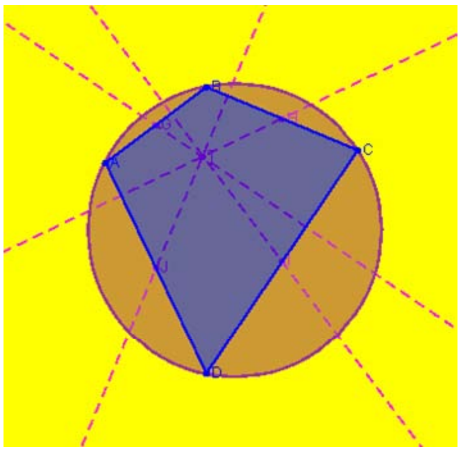

Fig. 21 Mathot's Point

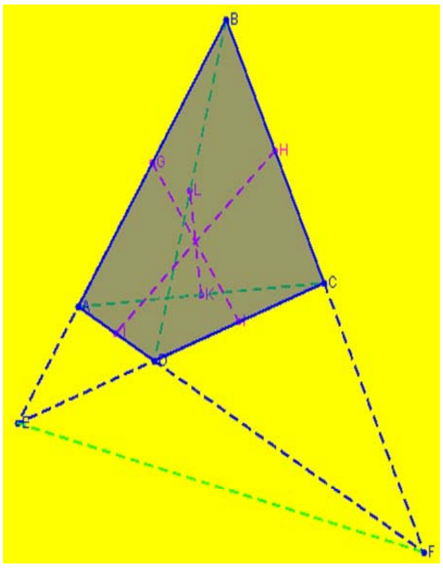

Fig. 22 Bimedians for a given quadrilateral

The implementation of the method which obtains bimedians for a given quadrilateral (Fig. 22) can be as follows:

```
public Segment2D[] bimediane() {
Segment2D[] bm=new Segment2D[3];
bm[0]=new Segment2D(sg[0].mijloc(),
sg[2].mijloc());
bm[1]=new Segment2D(sg[1].mijloc(),
sg[3].mijloc());
bm[2]=new Segment2D(diagonala(V[0]).mijloc(),
diagonala(V[1]).mijloc());
return bm; }
```

The Newton Line is draw by the following method of the class Patrulater2D (Fig. 23):

```
public Dreapta2D dr_Newton() {
return bimediane()[2].getSuport(); }
```

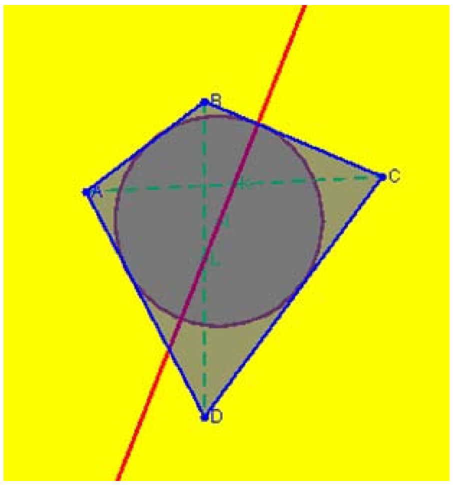

Fig. 23 Newton Line

The method of class Patrulater2D which obtains the Gauss Line (Fig. 24) is presented forwards:

```
public Dreapta2D dr_Gauss() {
return new Dreapta2D(diagonala(V[0]).mijloc(),
diagonala(V[1]).mijloc()); }
```

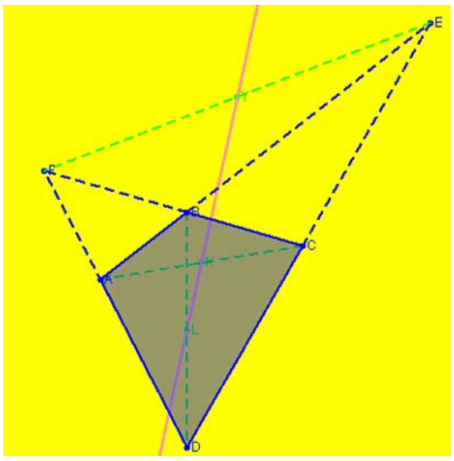

Fig. 24 Gauss Line

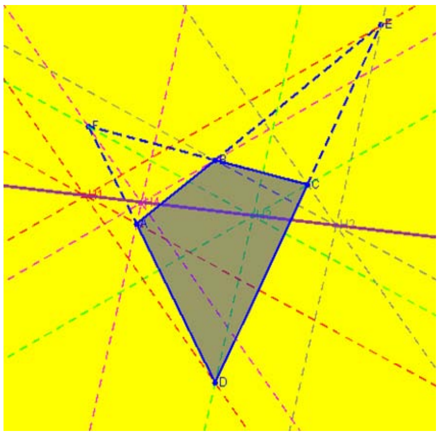

Fig. 25 Aubert Line for a convex quadrilateral

The implementation of the method which obtains Aubert Line for a quadrilateral (Fig. 25, Fig. 26) can be as follows:

```
public Dreapta2D dr_Aubert() {
Triunghi2D t1=new Triunghi2D(V[0],V[3],
diagonala3().getExtremitate1());
Triunghi2D t2=new Triunghi2D(V[0],V[1],
diagonala3().getExtremitate2());
return new Dreapta2D(t1.H(),t2.H()); }
```

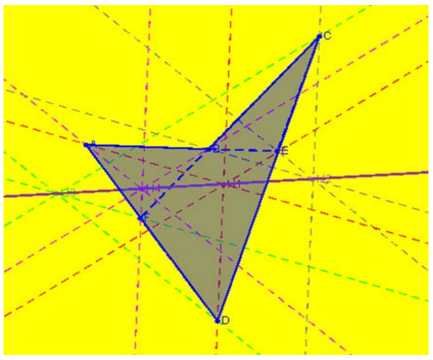

Fig. 26 Aubert Line for a concave quadrilateral

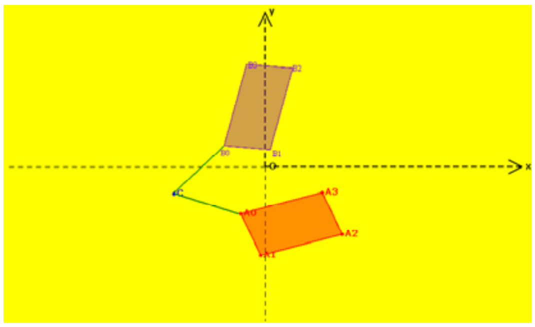

Fig. 27 Rotation of a parallelogram

The method of class Izometrie2D which obtains the rotation of a parallelogram (Fig. 27) is presented forwards:

```
Paralelogram2D izometrie(Paralelogram2D P) {
Punct2D[] A=new Punct2D[3];
for (int i=0;i<3;i++)
A[i]=new Punct2D(izometrie(P.getVarf(i)));
return new Paralelogram2D(A[0],A[1],A[2]); }
```

The method of class Omotetie2D which obtains the homothety of a pentagon (Fig. 28) is presented forwards:

```
public Poligon2D omotetie(Poligon2D P) {
int n=P.getNrVarf();
Punct2D[] A=new Punct2D[n];
for (int i=0;i<n;i++)
A[i]=new Punct2D(omotetie(P.getVarf(i)));
return new Poligon2D(n,A); }
```

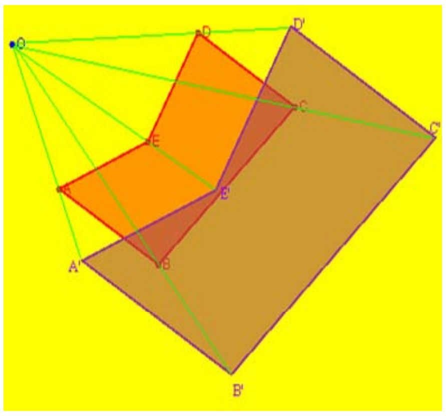

Fig. 28 Homothety of a pentagon

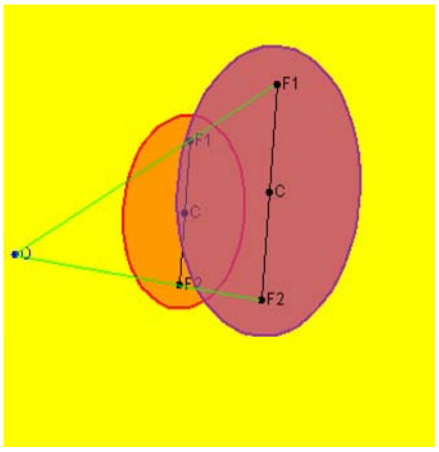

Fig. 29 Homothety of an ellipse

The method of class Omotetie2D which obtains the homothety of an ellipse (Fig. 29) is presented forwards:

```
public Elipsa2D omotetie(Elipsa2D E) {
Punct2D P1=new Punct2D(omotetie(
E.getFocar1()));
Punct2D P2=new Punct2D(omotetie(
E.getFocar2()));
Elipsa2D E1=new Elipsa2D(P1,P2,Math.abs(raport)
*E.getA());
Punct2D P[]=new Punct2D[73];
int i;
for (i=0;i<=72;i++) {
            P[i]=new   Punct2D(E.calcul_puncte()[i]);
            P[i].setare(omotetie(P[i]));
          }
E1.setare_puncte(P);
return E1;  }
```

3 Conclusion

The advantages of the presented software are:

- By the multitude of the offered facilities, this mathematic software can be used successfully for computer assisted training at geometry, both in the pre-university and university environment;
- The software is useful both in the step of acquiring new knowledges, the step of consolidating the acquired knowledge, and in the assessment step;
- Is achieved a high technical level, being taken into account all the methodical requirements;
- Are intended few exposure levels, depending on the schoolchildren's and students' preparation level;
- By the multitude of the offered options, the application can replace successfully the paper and the pencil for making the geometrical constructions.

Utilisation of the presented informatics system in studying geometry will contribute to build-up and develop the students' informational culture. The computer assisted training in the geometry elements' study process is also an efficient method to increase the learning motivation of this discipline and the quality of its acquiring.

References:

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

[2] E. Love, The functions of visualization in learning geometry, *Exploiting mental images with computers in mathematics education, Proceedings of NATO Advanced Workshop*, 1995, pp. 125-141, Berlin, Springer-Verlag

[3] E. Scalon, C. Tosunoglu, A. Jones, P. Butcher, S. Ross, J. Greenberg, Learning with computers : experiences of evaluation, *Computer Education*, Elsevier Science, 1998

[4] A. Mcdougall, A. Squires, Empirical study of a new paradigm for choosing educational software, *Computer Education*, Elsevier Science, 1995, Vol. 25

[5] R. Noss, C. Hoyles, Windows on mathematical meanings: Learning cultures and computers, *Mathematics Education Library*, Vol. 17, 1996, Dordrecht, The Netherlands

[6] D. Glusac, D. Radosav, D. Karuovic, D. Ivin, Pedagogical and Didactic-Methodical Aspects of E-learning, *6th WSEAS International Conference on E-ACTIVITIES*, Tenerife, Spain, December 14-16, 2007, pp. 67-75

[7] A.R. Lupu, R. Bologa, G. Sabau, M. Muntean, Integrated Information Systems in Higher Education**,** *WSEAS TRANSACTIONS on COMPUTERS*, Issue 5, Vol. 7, May 2008, pp. 473-482

[8] L.Y. Por, A.B. Zaitun, An Adaptive User Assessment Model for e-Learning**,** *WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION*, Issue 3, Vol. 5, March 2008, pp. 158-167

[9] C.E. Iglesias, A.G. Carbajo, M.A. Sastre Rosa, Interactive tools for Discrete Mathematics e-learning**,** *WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION*, Issue 2, Vol. 5, February 2008, pp. 97-103

[10] A. Ahmad, S.S. Salim, R. Zainuddin, A Cognitive Tool to Support Mathematical Communication in Fraction Word Problem Solving**,** *WSEAS TRANSACTIONS on COMPUTERS*, Issue 4, Vol. 7, April 2008, pp. 228-236

[11] D. Petcu, A. Eckstein, C. Giurgiu, Adapting a Legacy Code for Ordinary Differential Equations to Novel Software and Hardware Architectures**,** *WSEAS TRANSACTIONS on COMPUTERS*, Issue 5, Vol. 7, May 2008, pp.463-472

[12] C. Laborde, J.M. Laborde, What about a learning environment where Euclidean concepts are manipulated with a mouse?, *Computers and exploratory learning,* 1995, pp. 241-262, Berlin, Springer-Verlag

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

[14] R. Silveira, R. Viccari, JADE – Java Agents for Distance Education Framework, *8th Annual International Distance Education Conference,* January 23-26, 2001

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

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

[17] R. Pressman, Software Engineering - A Practitioners Approach, *McGraw-Hill*, 2005

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

[19] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, *Addison Wesley*, 1999

[20] D. Rosenberg, K. Scott, Use case Driven Object Modeling with UML, *Addison Wesley*, 1999

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