Geometry Lab with Mathematica

Mathematica is used in a course of Geometric Techniques for Computer Graphics & CAD

by Joan Trias and Vera Sacristán

Abstract

An experience of a Geometry Lab based on `Mathematica` for a Geometry course for Computer Science students, intended to be a basis for later courses in Computer Graphics, CAD and CAGD, is presented. Students combine their knowledge of basic geometry topics like polyhedra, curve and surface generation, affine and perspective transformations, among others, by means of lists manipulation and a clever use of substitution rules to generate apparently complicated geometric scenes in which geometric objects are placed in different positions in space. Several examples are offered, some of them together with their code. The reader can enjoy himself solving the proposed examples and producing new variants.

Introduction

We are currently teaching a course for Computer Science students (Facultat d'Informàtica de Barcelona, Universitat Politècnica de Catalunya, Barcelona) dealing with **geometric computation techniques** as a basis for later courses in the fields of Computer Graphics, CAD and CAGD.

We have designed and organized a Geometry Lab for this course based on `Mathematica` as a software tool ([3]); the aim of this contribution is to describe this project and the corresponding experience. A very short communication related to it has been presented in [1].

Describing the experience

The course is intended to cover different topics that are useful for computer graphics and computer aided design; among others, some of them are: polyhedra, generation of curves and surfaces, affine bi- and three-dimensional transformations, perpective transformations, elementary methods of curve and surface design in CAD, geometric algorithms. During the course our students must and also achieve what we could consider a certain maturity in three-dimensional geometry as well as in geometric objects manipulation.

More than presenting a classical course covering these subjects, we have preferred to teach students the **basic geometric operations** they will need while developing geometric applications in computer graphics and related fields. We have identified some of these operations; namely we want our students to be able to perform the following ones:

- Place geometric 3D objects onto a plane, such as in figure 1.

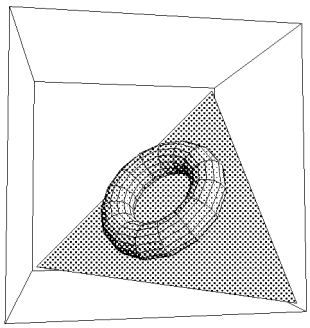

Figure 1. *Torus generated by* `shapes` *that has been put in tangent position onto the plane* $x + y + z - 3 = 0$.

- Draw 2D objects (curves, polygons or similar) onto a plane, for example onto the faces of some regular polyhedron (as can be seen in figure 2).
- Construct geometric objects that are in some sense defined by an "axis" or at least the position of which in space can be essentially determined by the position of an axis, as would be for example in the case of a cylinder, a cone, an helicoid o a torus (the axis we can consider in the last case being the normal to the diametral plane, the problem illustrated

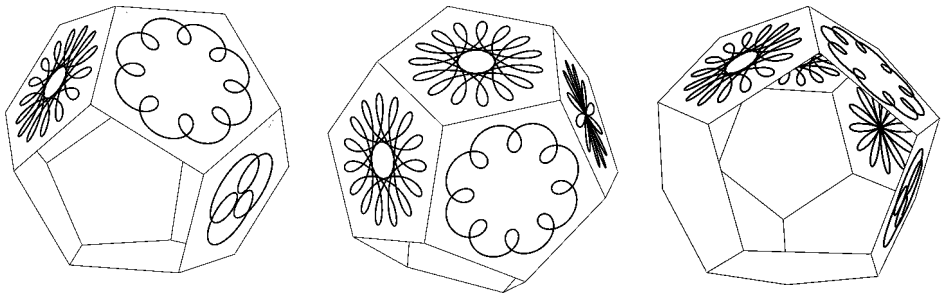

Figure 2. *Rhodonea, epitrochoids and other classical "mechanical" or "spirographical" curves that have been put on several faces of a dodecahedron generated by* `Polyhedra`.

in figure 1 can be reformulated in terms of an "axis" problem). Figure 3 shows an example of this kind of problem.

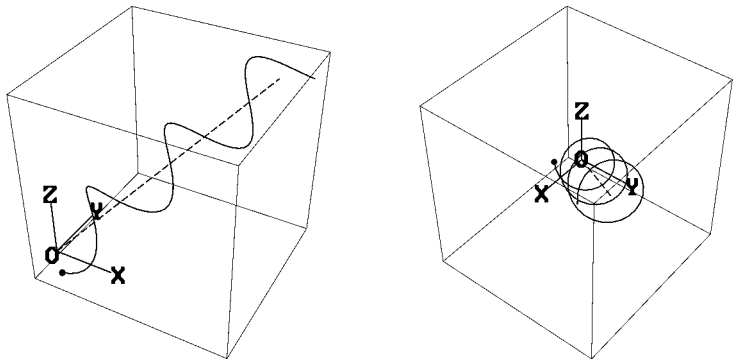

Figure 3. *Constructing a circular helix around an axis determined by the origin and the point* $(1, 1, 1)$.

Another problem related to that one could consist of constructing a circle with its center in a curve and placed onto the plane perpendicular to the curve at that point. This can be a point of departure for the synthesis of tubular surfaces.

- Produce the most common affine transformations; although many problems can be solved directly by changing the coordinate systems, students must know the specific language of transformations and use it to solve geometric problems, as for example in figure 4.

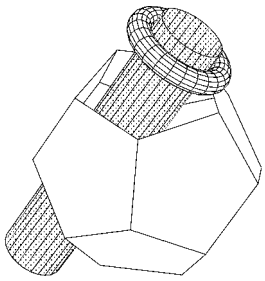

Figure 4. *A cylinder in original position has been transformed to produce another copy of it with an axis now determined by the baricenters of the two missing faces of the dodecahedron.*

As can be seen, our course has a highly practical orientation, that makes necessary a strong activity in a geometry lab. The examples we have seen are typical exercices we propose to our students.

The main tools

We distinguish between **geometric** and **programming** tools.

a) Geometric tools

From the point of view of **mathematics**, and specifically of **geometry**, the required background is not much extensive; the main tool we sistematically use is the **coordinate transformation**, specially between cartesian coordinate systems.

To establish some notation, recall that if we consider two coordinate systems, $S = (O, \{\vec{e}_1, \cdots, \vec{e}_n\})$, with O being the origin and $\{\vec{e}_1, \cdots, \vec{e}_n\}$ the basis of the corresponding vector space, determining the coordinate axes, and a "new" coordinate system given by $S' = (O', \{\vec{e}_1\,', \cdots, \vec{e}_n\,'\})$, then the "old" coordinates X of a point, and the "new" ones X' of the same point are related, as is well known, by the matrix equation

$$X = AX' + W,$$

where $W \overset{(S)}{=} \overrightarrow{OO'}$ is the position vector of the new origin expressed in the system S, and A is the "change of basis matrix", formed with the column vectors of the new basis as linear combinations of the old one; here W,

X and X' are also column vectors. We can also derive the relation $X' = A^{-1}(X - W)$. We shall refer to this transformation as the "inverse" one, while the former version will be called the "direct" one.

Students can program constructions of complex three-dimensional scenes with a sistematic and clever use of coordinate transformations, making a proper choice of the coordinate systems involved, sometimes concatenating and sometimes combining direct and inverse changes. A typical example of use of direct and inverse transformations can be seen in figure 5, corresponding to a variant of one of the exercices our students must solve. Seemingly complicated efects are not difficult to generate.

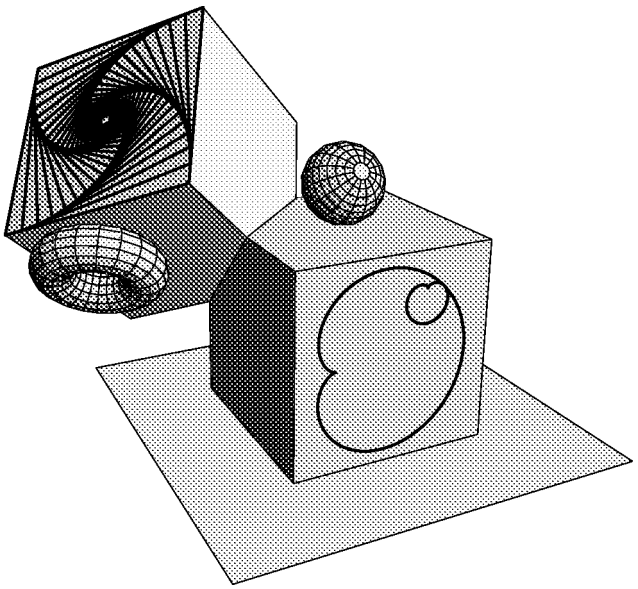

Figure 5. *Observe the cardioid with an inner one, both tangent at a common point.*

Our main tool is very simple, and with such a simple tool we are able to go very far, from the point of view of the complexity of geometric scenes our students are able to produce.

b) Programming tools

From the point of view of **programming**, there are many reasons that explain why we use `Mathematica`; let us mention, among others, the fo-llowing ones:

- Proximity to mathematical language, particularly matrix and vector calculations, with a simple notation.
- Extensive supply of geometric objects, as well as incorporated functions to produce geometric objects such as curves or surfaces.
- Powerful 3D graphics processing.
- Powerful manipulation of lists.
- Powerful and precise substitution rules.

Let us discuss in the sequel the before mentioned items.

Proximity to mathematical language allows us to write `Mathematica` functions almost as if we were writing mathematical functions or expressions; consider for example the definition of norm derived from the dot product:

```
norm[v_]:=N[Sqrt[v.v]]
```

or the definition of cross product:

```
e1={1,0,0}
e2={0,1,0}
e3={0,0,1}
vectorProduct[u_,v_]:=
 N[{Det[{u,v,e1}],Det[{u,v,e2}],Det[{u,v,e3}]}]
```

The *richness of geometric objects*, ranging from the most elementary ones (geometric primitives like `Polygon`, `Line`, `Point`, and others) to more complex ones, allows us to generate non trivial objects. We also make a heavy use of the packages `Shapes` and `Polyhedra` and, to a minor extent, of `SurfaceOfRevolution`, as well as of the incorporated functions to produce curves or surfaces.

Powerful 3D graphics processing is important because it allows our students to concentrate on geometry, which is the objective of the course: once the plan to solve a problem is traced from the point of view of geometry, then

Mathematica does the rest, specially 3D rendering and perspective projections, so that they don't need to pay attention to such details, important, but lengthy to solve.

Powerful manipulation of lists allows to easily manipulate geometric structures, due to the list structured coding of geometric objects and scenes; the possibility of coding complex geometric scenes by means of lists renders easy their manipulation and transformation, and permits cumulative construction from simple to more complex structures.

As an example, load the package shapes (and suppose it loaded from now on) and generate a cylinder cyl0 with

```
Needs["Graphics`Shapes`"]
cyl0=Cylinder[]
Show[Graphics3D[cyl0]]
```

Observe that what we have really obtained is a list of polygons, which are the quadrangular faces of the lateral surface by means of which we approximate the true cylinder by a polyhedral surface; indeed, we have cyl0 equal to

```
{
 Polygon[{{0.951056516295153, 0.3090169943749474, 1},
         ...
         {0.809016994374947, 0.5877852522924732, 1}}],
 ...
 Polygon[{{1., 0, 1}, {1., 0, -1},
          {0.951056516295153, 0.3090169943749474, -1},
          {0.951056516295153, 0.3090169943749474, 1}}]
}
```

It is not difficult now to manipulate this structure having in mind that manipulating lists produces geometric manipulation of the scenes (figure 6):

```
cyl00=Drop[cyl0,{2,5}]
Show[Graphics3D[cyl00],Boxed->False,ViewPoint->{2,2,2}]
```

Perhaps one of the **most important** characteristics of Mathematica for us is the possibility to easily program *selective substitutions* inside a complicated

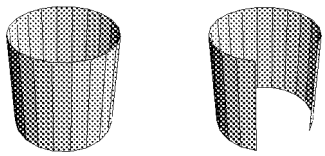

Figure 6

tree-like structure describing a geometric object or a geometric scene; this allows us to easily program transformations and in general manipulations of geometric objects, changing only the coordinates, but retaining the overall structure.

Let us consider an example; suppose we want to transform the cylinder `cyl0` by $(x, y, z) \mapsto (2x, 3y, z)$; just write a substitution rule and look at the result by representing both objects in the same scene (figure 7):

```
cyl1=cyl0/.{x_?NumberQ,y_?NumberQ,z_?NumberQ}->{2x,3y,z}
Show[Graphics3D[{cyl0,cyl1}]]
```

Now we could apply a new transformation to the scene `scene={cyl1,cyl0}`, such as for example to interchange the roles of the axes z and x; it suffices to use a substitution rule (see the result in figure 7).

```
scene2=scene/.{x_?NumberQ,y_?NumberQ,z_?NumberQ}->{z,y,x}
Show[Graphics3D[scene2]]
```

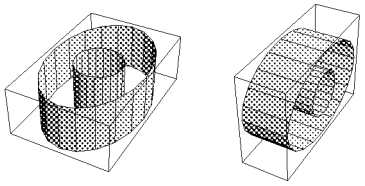

Figure 7

With these possibilities of substitution rules, the **point** transformation $X = AX' + W$ can be easily enlarged to produce the transformation of a **whole geometric object** or even a **complete geometric scene** formed by several objects (with a slightly more robust formulation even graphic objects can be transformed); this means that a complex object or scene can be expressed, as a whole, in an another coordinate system, with no need of decomposing or taking apart the elementary geometric components of the complex object, express each of them in the old or new coordinate system, and finally reconstruct the whole structure, as had to be done in a more classic programming

environment. That is the point that makes this method so powerful and easy at the same time, in order to perform geometric transformations.

The most simple version for a function transforming new coordinates into old ones could be the following one, in the 3D case:

```
coordinateTransf[(*X=AX'+W*)
 obj_, (*object or scene*)
 A_,   (*change of basis (row) matrix*)
 W_    (*new origin*)
]:=N[obj]/.{a_?NumberQ,b_?NumberQ,c_?NumberQ}->A.{a,b,c}+W
```

This function can be trivially completed to cope with the "inverse" coordinate transformation, in the case where both vector bases are orthonormal, and even in other cases, and also enhanced from the programming point of view; for the sake of simplicity, we express it in this form.

With these tools in hand we are able to challenge our students with sofisticated (and, we expect, imaginative) 3D geometric scenes.

Let us present now the general strategy of resolution of geometric problems consisting of scenes or objects we wish to produce using coordinate transformations, by means of the resolution of some examples.

A basic example

Let us begin with a simple problem: drop a circular cylinder `cylA` of radius 1, height 1, onto the plane $\pi : x + y + z = 3$ such that the cylinder's axis passes through the point $P = (1, 1, 1) \in \pi$. First of all, generate such a cylinder in canonical position, that is with its symmetry axis coincident with coordinate axis Oz and the base onto the plane $z = 0$:

```
cylA=ParametricPlot3D[{Cos[t],Sin[t],z},{t,0,2Pi},{z,0,1},
     PlotPoints->{20,2},DisplayFunction->Identity][[1]]
```

This will be a 3D object precomputed once and for all; we thus obtain the coding not only of this cylinder but of all the cylinders with the same metric characteristics in any other cartesian system of coordinates $(O'; \{x', y', z'\})$,

provided that its position mimics exactly the position of the canonical one, that is, its axis coincides with the third axis of coordinates $O'z'$ and its base is on the coordinate plane $z' = 0$. This is the basic idea for our strategy of resolution of this kind of problems: suppose the problem solved and the cylinder put onto the plane in the right position; place an appropiate cartesian system of coordinates, related to the plane and to the cylinder: then the description of the object on top of the plane expressed in this system of coordinates is exactly the canonical one; hence it only remains to express it in the old cartesian system. So, the only problem is to construct the new cartesian system so that the third axis $O'z'$ coincides with the axis of the cylinder and such that $z' = 0$ is identified with the plane π.

Let us construct such a basis manually (it can be automatized):

```
uu3={1,1,1} (*normal to the plane*)
u3=uu3/norm[uu3]
uu1=vectorProduct[u3,e3] (*for example, if not dependent*)
u1=uu1/norm[uu1]
u2=vectorProduct[u3,u1]
AA={u1,u2,u3}
A=Transpose[AA]
```

Now, with `W={1,1,1}`, we can consider the new coordinate system and express the cylinder in the ancient one:

```
plane=Polygon[{{3,0,0},{0,3,0},{0,0,3}}]
axes[length_:1]:={Line[{{0,0,0},{length,0,0}}],
                  Line[{{0,0,0},{0,length,0}}],
                  Line[{{0,0,0},{0,0,length}}]}
axesOnThePlane=coordinateTransf[axes[],A,W]
cylinderOnThePlane=coordinateTransf[cylA,A,W]
Show[Graphics3D[{axes[3.2],plane,axesOnThePlane}],
Boxed->False]
Show[Graphics3D[{axes[3.2],plane,cylinderOnThePlane}],
Boxed->False]
```

and see the result in figure 8.

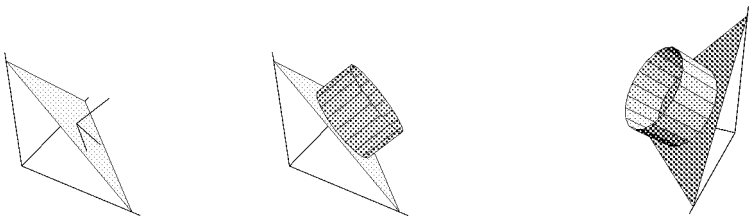

Figure 8

A more complete example

Let us now consider a more complex scene (figure 9), composed by part of a cylinder generated by `shapes`, of radius 2, height 8, with axis coincident with Oy, an icosahedron obtained with `Polyhedra`, reescaled and with center at $(3.5, -1, 2.5)$ (by means of a coordinate transformation) and three spheres obtained with `shapes` with centers, respectively, at $(0, 0, 0)$, $(-3, 0, 0)$ and $(0, 0, 3)$, that are "looking at" a point, the center of the polyhedron, that is, with their polar axes pointing to that point.

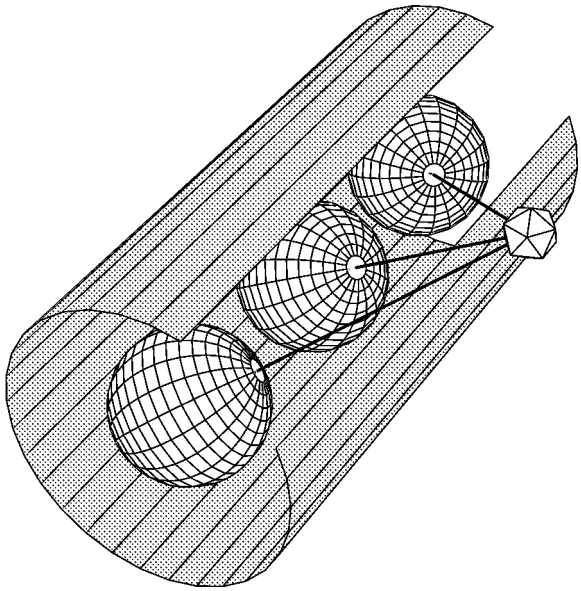

Figure 9

This can be obtained by the following code (where we have slightly automa-

tized the process of contructing an orthonormal basis, in which the third axis is given by two points supposed to be different and is also supposed not to be parallel to the Oz axis):

```
Needs["Graphics`Polyhedra`"]

basis[A_,B_]:=Module[{uu3,uu1},
     uu3=B-A; (*supposed different*)
     u3=uu3/norm[uu3];
     uu1=vectorProduct[u3,e3]; (*supposed independent*)
     u1=uu1/norm[uu1];
     u2=vectorProduct[u3,u1];
     Transpose[{u1,u2,u3}]]

lookAt={3.5,-1,2.5}

icosa0=Icosahedron[]
scale={{.3,0,0},{0,.3,0},{0,0,.3}}
icosa1=coordinateTransf[icosa0,scale,lookAt]

cyl0=Cylinder[2,4,30]
cyl1=Drop[cyl0,{2,5}]
cyl2=cyl1/.{x_?NumberQ,y_?NumberQ,z_?NumberQ}->{x,z,y}

sph0=Sphere[1,20,20]

center1={0,0,0}
sph1=coordinateTransf[sph0,basis[center1,lookAt],center1]
L1=Line[{center1,lookAt}]

center2={0,-3,0}
sph2=coordinateTransf[sph0,basis[center2,lookAt],center2]
L2=Line[{center2,lookAt}]

center3={0,3,0}
sph3=coordinateTransf[sph0,basis[center3,lookAt],center3]
L3=Line[{center3,lookAt}]
```

```
Show[Graphics3D[{
     {GrayLevel[.7],cyl2},
     {GrayLevel[1],sph1,sph2,sph3,icosa1},
     {L1,L2,L3}
     }], Boxed->False, PlotRange->All, Lighting->False]
```

Now we are ready to place the new object just created into any other scene, in any desired position (as onto a plane, like in figure 10).

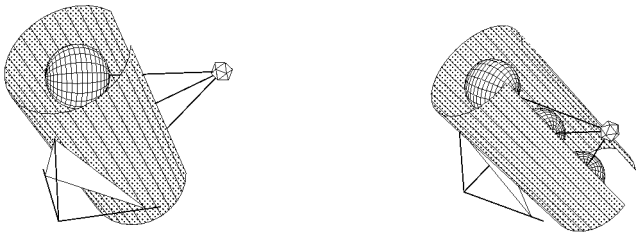

Figure 10

Some other examples

There are many additional programming exercices we propose to our students; the complete set of them for our Geometry Lab can be found in [2].

Let us mention only a sample of them.

We sistematically use `ParametricPlot` and `ParametricPlot3D` to generate curves and surfaces, usually in general position in space; consider for example figure 11.

Concerning 2D affine transformations, an exercise incorporating a great number of different transformations is the construction of a **toothed wheel** from the description of its smallest piece: the tooth. After the programming task, prisms based on the wheel can be generated and used as a component in more complex scenes, concerning now 3D affine transformations, such as for

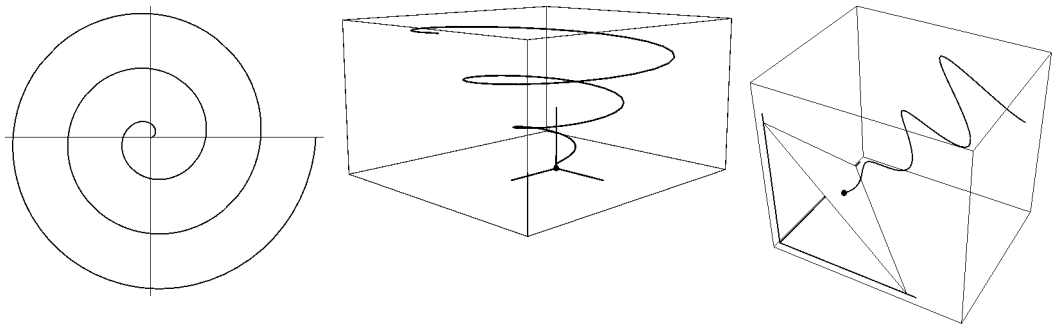

Figure 11. *Constructing a spiral helix, with plane projection onto an Archimedes spiral, and then putting it in a precise position onto the plane* $x + y + z = 3$.

example specular symmetry (see figure 12).

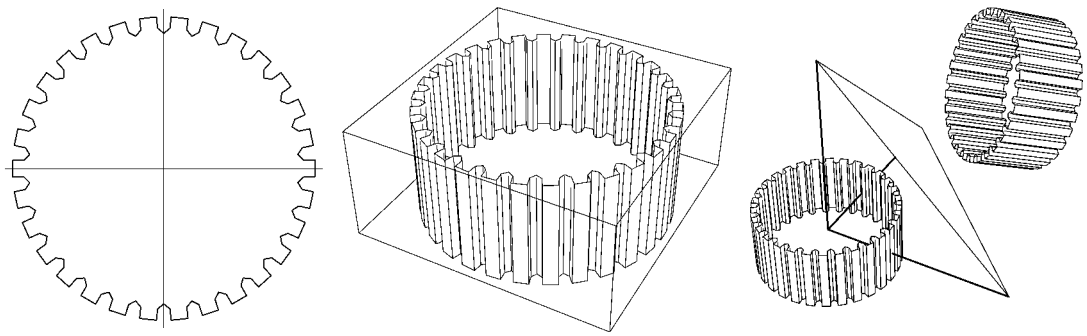

Figure 12. *Toothed wheels, prisms and specular symmetry.*

Another interesting exercise is the generation of **polyhedra of revolution** from a plane polygonal profile, as a standard tool for volume generation in solid modelling and CAD; one of the most popular objects is the **chess queen**, that we also integrate in more complex geometric compositions, such as in figure 13.

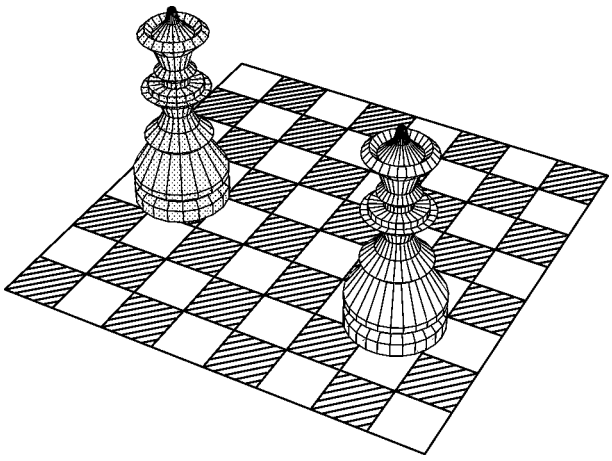

Figure 13. *Composition with chess pieces of revolution.*

Conclusions

Our geometry lab is not merely an experimental complement in our course, but it is an integral part of it, and one of the most important ones. Without the possibility of using `Mathematica` we would not have been able to organize the course as we have done, and without it our course would have been very different.

Our impression is that students learn much better geometry with such kind of lab exercices, they become enthousiastic about them and they realize the importance of the relationship between theory and applications.

With our experience, we hope we have contributed to make many people more familiar with mathematics laboratories, specially geometry ones, and to be helping that way to the incorporation of the new developing technologies in mathematical education, to put the teaching of mathematics in a new high-tech environment, and consequently, on a new and modern basis.

As a byproduct of our preparation of materials for the course we have at our disposal an extensive set of `Mathematica` functions that constitute a powerful tool for mathematical calculation and illustration (specifically, in the difficult field of 3D geometrical illustration); once it will be organized as a `Mathematica` package, it will become available. By now, our contribution to this use of `Mathematica` in preparing graphic material for teaching can be found in [2]. Some other authors in our Department have developed

`Mathematica` packages for geometrical calculation and have used them as an illustration tool for educational purposes to prepare classroom material ([4]).

References

[1] Hurtado, F., Trias, J.: *Técnicas geométricas para la informática gráfica: prácticas de laboratorio,* Actas de las Jornadas sobre Nuevas Tecnologías en la Enseñanza de las Matemáticas en la Universidad (Eds. A.Montes, J.M.Brunat), TEMU-95, (241-250), Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, 1995.

[2] Trias, J.: *Geometria Computacional, pràctiques de laboratori*, Universitat Politècnica de Catalunya, 1995.

[3] Wolfram, S.: `Mathematica`*, a system for doing mathematics by computer*, Addison-Wesley, 1991.

[4] Xambó, S., Grané, J.: *Geometría con* `Mathematica`*,* Actas de las Jornadas sobre Nuevas Tecnologías en la Enseñanza de las Matemáticas en la Universidad (Eds. A.Montes, J.M.Brunat), TEMU-95, (425-457), Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, 1995.

About the authors

Both authors are mathematicians and Professors of Mathematics in the Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Barcelona, and are teaching courses in Computational Geometry.

Their research interest is also centered in the field of Discrete, Combinatorial and Computational Geometry (complexity theory and algorithm design).

Both are enthousiastic about the use of advanced mathematical software for classroom laboratories, as a valuable aid in teaching mathematics.

Joan Trias
Dept. de Matemàtica Aplicada II
Facultat d'Informàtica de Barcelona
Pau Gargallo, 5
Universitat Politècnica de Catalunya
E-08028 Barcelona, Spain
e-mail: avtrias@ma2.upc.es

Vera Sacristán
Dept. de Matemàtica Aplicada II
Facultat d'Informàtica de Barcelona
Pau Gargallo, 5
Universitat Politècnica de Catalunya
E-08028 Barcelona, Spain
e-mail: vera@ma2.upc.es