Three-dimensional graphics with TikZ/PSTricks and the help of GeoGebra

Luciano Battaia

Abstract

In this article we consider the opportunity of using dynamic geometry software, such as GeoGebra, to allow easy exporting of three-dimensional geometric pictures, with subsequent 2D parallel projection, in PGF/TikZ or PSTricks code. The help of software like GeoGebra considerably simplifies the production of very complex pictures in LaTeX code, requiring only a basic knowledge of PGF/TikZ or PSTricks languages and taking advantage of a substantially mouse driven program. All examples and sample code here are in PGF/TikZ, but almost nothing changes if one prefers PSTricks.

1 Introduction

LaTeX users, particularly those writing scientific papers, have always had a need for high-quality vector graphics, including labels, that fit the style of the rest of their documents.

There is no special problem in the case of two-dimensional graphics, and the two most widespread tools PSTricks and PGF/TikZ (that from now on will only be mentioned merely as TikZ), together with their derived packages, solve almost every problem very well. As Claudio Beccari has shown [2], LaTeX's basic *picture* environment is sufficient for many situations.

Things change substantially if we are interested in three-dimensional graphics. Plots of two-variable functions and of various kinds of surfaces can easily be handled using dedicated packages, for instance pst3dplot or pgfplots. Also, for geometric figures some very interesting packages are available, for example pst-solides3d or pst-3d in the PSTricks family or tikz-3dplot in the TikZ family, but in all cases a rather deep knowledge of programming techniques in PSTricks or TikZ is needed and, in our opinion, this is not at all easy for the average user.

External programs that produce PSTricks or TikZ codes can help, for instance *Sketch* by Eugene Ressler (see for example [3]) and *TEXgraph* by Patrick Fradin (`texgraph.tuxfamily.org/`). The last one in particular is very powerful and can also produce *POV-Ray* code, but, again, it is not within the reach of most users. Almost the same remarks apply to *Asymptote*, whose code can be directly included in a LaTeX source through the asymptote package.

An interesting and detailed introduction to the problem of producing three-dimensional graphics with TikZ can be found in an article by Keith Wolcott [5]. It was in fact the reading of this article that led us to study the problem in order to find a more accessible solution. Wolcott's article ends with a figure which shows only the partial solution of what had been the main purpose of the project: the drawing of two spheres and their circle of intersection. The author himself points out that the figure needs more work.

This is the reason why we begin this article with figure 1, which exactly reaches Wolcott's goal. Explanations on how we obtain it will be given later, but we immediately point out that our approach to the problem is completely different from Wolcott's.

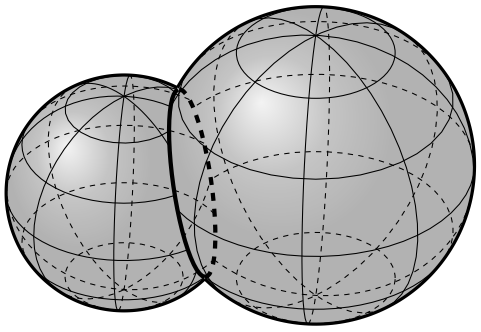

Figure 1: The intersection of two spheres with the circle of intersection

For the sake of completeness, we mention that a slightly different version of this article, in Italian, can be found in [1].

2 The coming of GeoGebra on the scene

For educational reasons we have been using *GeoGebra* for a long time, both because its non-commercial use is free and because its basic use is extremely simple. With reference to the problem we are dealing with, the interesting thing is the possibility of producing complex two-dimensional figures and exporting them in PSTricks or TikZ code, that can then be copied and pasted directly into a LaTeX source with only very limited adaptations, mainly regarding correct label positioning. The required knowledge of LaTeX packages is minimal and manageable for even inexperienced users. In short, anyone can produce even complex figures to be included in LaTeX documents with a WYSIWYG technique and extensively using the mouse. This seems far from what a LaTeX user normally does, but we think that in the case of graphics this strategy is preferable for many users. Of course one must know GeoGebra well enough, but this does not require the study of long and complex handbooks and, at any rate, dynamic geometry

software is of great help in experimenting with the construction of technical figures. An example of a complex 2D figure easily produced in GeoGebra and exported into Ti*k*Z almost without intervention in the generated code is shown in figure 2.

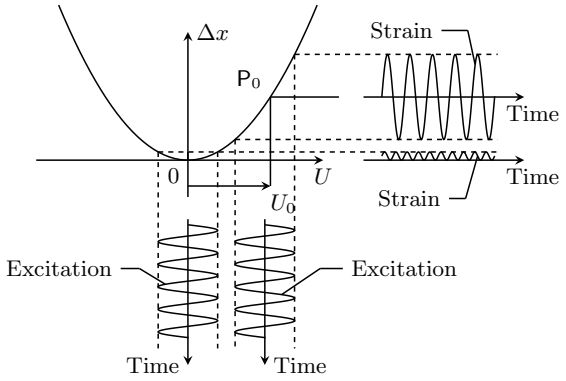

Figure 2: A picture produced with GeoGebra and exported into Ti*k*Z, used for a master's degree thesis

GeoGebra can export figures into both PSTricks and Ti*k*Z code (and even into *Asymptote*); in this article we consider only the case of Ti*k*Z, with which all the figures shown are realized. However, as mentioned above, substantially nothing changes if you prefer PSTricks, because, apart from some adaptations and some limited work to clean up the code, everything is automatically produced. For this reason also, only a few fragments of source code will be included. In addition, it should be noted that the generated code is not very interesting, as it consists almost exclusively of `\draw` instructions; all needed calculations have already been done by GeoGebra.

Some time ago a new version of GeoGebra (Geogebra Classic 5.0) which supports three-dimensional graphics was released. Unfortunately, for this 3D version no export into a LaTeX format has yet been implemented and, in our opinion, this will not be possible, at least not in a reasonably short time. Because of this limitation we decided to experiment with the possibility of directly executing a 3D to 2D projection in GeoGebra and then exporting it into Ti*k*Z code. Indeed, each 3D figure is just an appropriate 2D projection of a three-dimensional object.

Keeping this in mind, the first thing we tried to reproduce is a sphere originally drawn by Tomasz M. Trzeciak [4]; it was also reproduced by Keith Wolcott in [5]. Please compare Trzeciak's original (figure 3) with ours (figure 4).

The two pictures are almost identical but the Ti*k*Z codes are indeed completely different; you can compare them in detail in the Italian version of this article [1]. Here we only want to point out the fact

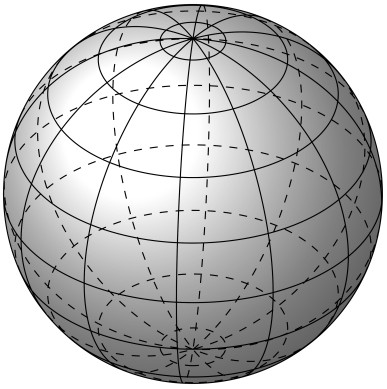

Figure 3: Sphere with meridians and parallels, produced by Tomasz Trzeciak using PGF/Ti*k*Z

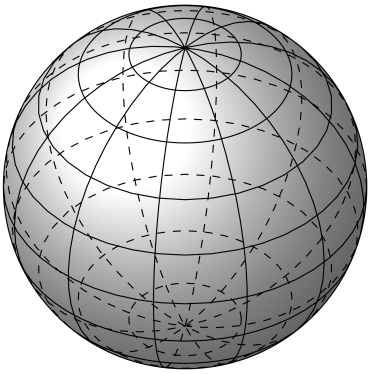

Figure 4: Sphere with meridians and parallels, produced with code exported from GeoGebra

that Trzeciak's code is much more concise and elegant, but it requires a deep knowledge of PGF programming. In fact you must first instruct PGF to make the correct calculations for the visible and invisible parts of each latitude or longitude circle, using appropriate PGF macros, and only after that you can draw the circles. In our code all calculations are made by GeoGebra, and only the drawing part is left to Ti*k*Z.

3 Some maths behind the scene

GeoGebra is a very well structured and powerful program for dynamic geometry. There are two different windows for 2D graphics, a window for 3D graphics, a fairly complete spreadsheet, a probability calculator and an algebra window where you can read the coordinates of the points, the equations of the curves, and so on. The very important feature is that all the windows can interact with each other. Regarding our problem, all that is obtained in the 3D window can be appropriately transferred to the main 2D window

(and then exported into LaTeX code). Moreover, it is interesting to note that GeoGebra is in any case "LaTeX oriented"; all textual annotations are inserted in the windows with LaTeX code.

Let us consider a Cartesian orthogonal system in three-dimensional space, that in GeoGebra is displayed in the 3D window, with an upward vertical z-axis; call α a rotation around the vertical axis and β a rotation around a horizontal axis. The parallel projection of this Cartesian system in a plane (that in our case will be the main 2D window of GeoGebra), can be obtained, for instance, with the following formulas:

$$\begin{aligned}\vec{i} &= \big(-\cos(\alpha), -\sin(\alpha)\sin(\beta)\big)\\ \vec{j} &= \big(\sin(\alpha), -\cos(\alpha)\sin(\beta)\big)\\ \vec{k} &= \big(0, \cos(\beta)\big)\end{aligned} \quad ,$$

where $\vec{i}$, $\vec{j}$, $\vec{k}$ are the vectors of the basis. If you set the origin to the point $\mathsf{O} = (0, 0)$, which is preferable, you must create two angle sliders with the names α and β; afterwards the basis vectors can be constructed with the following GeoGebra code:

$$\begin{aligned}\mathrm{i} &= \mathrm{Vector}[\mathsf{O}, (-\cos(\alpha), -\sin(\alpha)\sin(\beta))]\\ \mathrm{j} &= \mathrm{Vector}[\mathsf{O}, (\sin(\alpha), -\cos(\alpha)\sin(\beta))]\\ \mathrm{k} &= \mathrm{Vector}[\mathsf{O}, (0, \cos(\beta))]\end{aligned} \quad .$$

Now, if you consider a point $\mathsf{P} = (x_\mathsf{P}, y_\mathsf{P}, z_\mathsf{P})$ in the 3D window, its projection will be

$$\mathsf{P}' = x_\mathsf{P}\,\vec{i} + y_\mathsf{P}\,\vec{j} + z_\mathsf{P}\,\vec{k},$$

or, in the language of GeoGebra,

$$\mathsf{P}' = x(\mathsf{P})\,\mathrm{i} + y(\mathsf{P})\,\mathrm{j} + z(\mathsf{P})\,\mathrm{k}.$$

If you consider instead a curve C with parametric equations $(f(t), g(t), h(t))$, with the parameter t appropriately included between two extremes, its 2D projection, always in the GeoGebra language, will be

$$\begin{aligned}x' &= f(t)\,x(\mathrm{i}) + g(t)\,x(\mathrm{j}) + h(t)\,x(\mathrm{k})\\ y' &= f(t)\,y(\mathrm{i}) + g(t)\,y(\mathrm{j}) + h(t)\,y(\mathrm{k})\end{aligned} \quad .$$

These formulas allow the 2D projection of every figure made in the 3D window of GeoGebra. After that you can experiment to find the best view for the figure by changing the angles α and β, working in the 2D window; this is an important feature because in general it is very difficult to find the appropriate viewing angle, and only trying over and over again can lead to the solution. Naturally not even GeoGebra minimizes the problem of 3D graphics as it is clear that those who need images of this type must have a good mathematical preparation. Nothing is obtained for free!

In light of these formulas let's see in detail, as an example, how the sphere of figure 4 can be obtained. Begin by plotting the 3D sphere with center the origin and radius r and its 2D projection that is simply the circle with center the origin and again radius r. Next draw the parallels and meridians simply intersecting the sphere with appropriate planes. If, for instance, you need five parallels they will be found at the latitudes $-60°$, $-30°$, $0°$, $30°$, $60°$ and the corresponding planes have the following equations

$$\begin{aligned}z &= r\sin(-60°)\ ;\ z = -r\sqrt{3}/2\\ z &= r\sin(-30°)\ ;\ z = -r/2\\ z &= r\sin(0°)\ \ ;\ z = 0\\ z &= r\sin(30°)\ \ ;\ z = r/2\\ z &= r\sin(60°)\ \ ;\ z = r\sqrt{3}/2\end{aligned} \quad .$$

These planes can be plotted simply by writing the equations in the input bar. Now ask GeoGebra to find the intersection circle of the planes with the sphere and choose (for instance using the mouse) five points on each circle. After projecting these points on the 2D window, plot the conic through them, using the specific Command; this will be the projected parallel. Do the same for the meridians. Now, after choosing the best viewing angle, highlight the visible and invisible part of each ellipse. For the invisible part you can decide if you want to show it or not, you can choose a broken line, a reduced thickness, and so on. When everything is perfectly configured, export into Ti*k*Z (or PSTricks) and insert the code in your LaTeX document; it usually works very well and only small adaptations are normally needed, for instance regarding the position of the labels or if you need special shading. The technique that we have illustrated is absolutely basic; with a little experience in the use of GeoGebra, everything can be faster and further automated.

A point that deserves further attention from what has been previously described is how to treat the visible or invisible parts of the projected figure in GeoGebra. The 3D window of the software can automatically handle the visible or invisible parts, as shown in the screen shot of figure 5.

The projection of this picture on the 2D window produces an image where visible and invisible parts are plotted in the same style, as shown in figure 6; such a figure can't be exported as it is.

Now, by comparing the side by side images of the 3D and 2D windows, and using the Intersect command of Geogebra, one can correctly highlight the visible and invisible parts of each curve and finally obtain the image ready to export. It is shown in figure 7.

Regarding the intersection of two spheres, plotted in figure 1, there are no further complications since the intersection circle can be found directly by

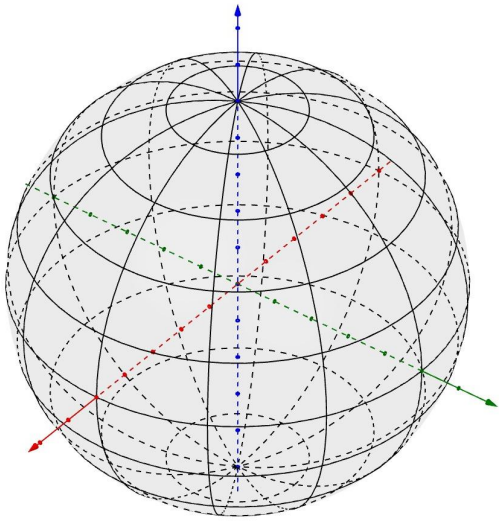

Figure 5: Screenshot of the main 3D window of GeoGebra for the production of the sphere of figure 4

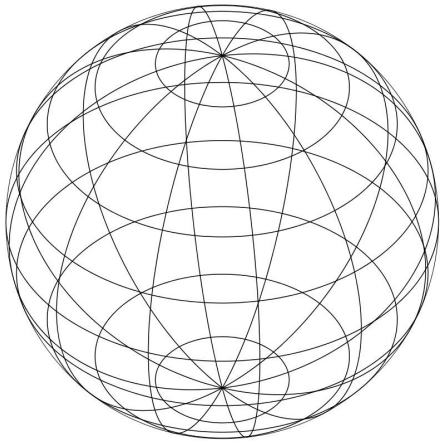

Figure 6: Screenshot of the 2D projection in Geogebra of the 3D window shown in figure 5

GeoGebra and then projected on the 2D window as described. In this case we have chosen not to show the overlapping parts of the spheres at all, in order to obtain a more readable figure.

4 A spiral on a sphere

The following example requires a minimum of extra mathematics, but no further work on the code. The goal is to plot a complex spiral, with endless turns, on a sphere, highlighting the property that the angle between the meridians and the spiral remains constant. The best way to solve the problem is to use

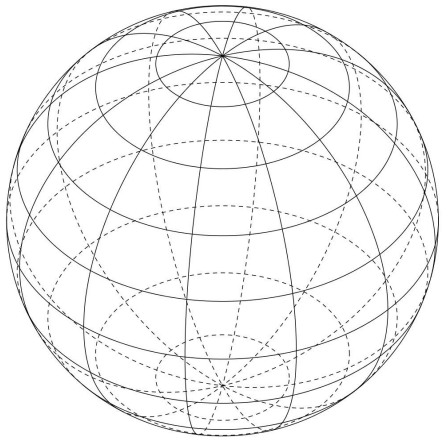

Figure 7: Screenshot of the main 2D window of Geogebra ready to be exported for the production of the sphere of figure 4

the following parametric equations of the spiral

$$\begin{aligned} x(t) &= \frac{r\cos t}{\sqrt{1+a^2t^2}} \\ y(t) &= \frac{r\sin t}{\sqrt{1+a^2t^2}}\,, \\ z(t) &= \frac{-art}{\sqrt{1+a^2t^2}} \end{aligned}$$

where r is the radius of the sphere and a is a parameter. It is preferable to set up r and a with sliders in GeoGebra and then choose the best values after testing different ones. The tracing of the tangent vectors and of the angles identified by them is straightforward.

The only thing that needs special attention is the fact that plots of lines such as the one needed here can't be drawn directly by Ti*k*Z and you need external software, for instance GNUPLOT, but this can be done in a straightforward way, and, in any case, GeoGebra automatically handles this problem in the export procedure! It should be noticed that PSTricks handles directly these situations. The final plot is shown in figure 8.

5 Polyhedra

One of the situations where GeoGebra's intervention is truly providential is the drawing of polyhedra and their developments; there are special routines to draw, in particular, Platonic solids and to show dynamically their development. The 2D projection of such figures is indeed very simple because you need only to find the correct position of the projected vertices, whose three-dimensional coordinates are automatically found by GeoGebra. Figure 9 shows the dodecahedron, while figure 10 shows a step towards its development in a plane.

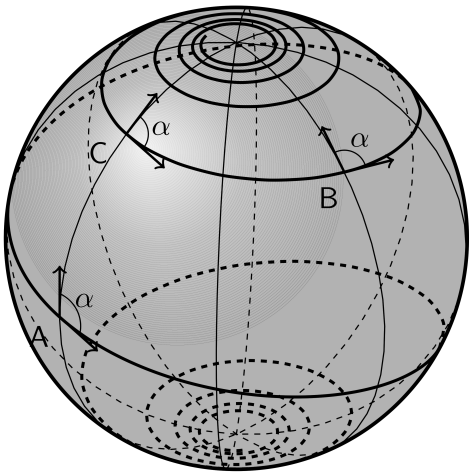

Figure 8: A spiral on a sphere

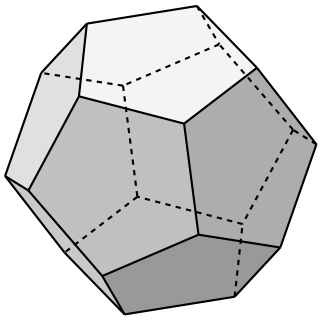

Figure 9: The regular dodecahedron

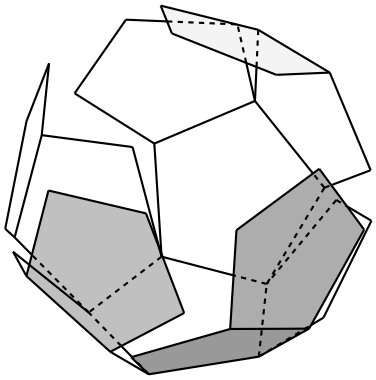

Figure 10: The regular dodecahedron: a step towards its development in a plane

Don't be fooled by the apparent simplicity of these pictures. The hand calculation of the coordinates of the vertices of the dodecahedron is not easy at all, and, even worse, their position during development!

Also, the drawing of the inscribed and circumscribed spheres is straightforward and you can see an example concerning the octahedron in figure 11.

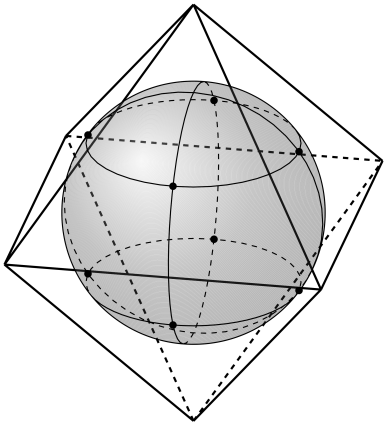

Figure 11: The regular octahedron and its inscribed sphere. The meridians and the parallels through four of the eight tangent points are highlighted

Even more important is the fact that the drawing of the curves described by the vertices during development is relatively straightforward. In GeoGebra every vertex can leave a track during the development and it is possible to project this track in the 2D window; it is now very simple to plot, using a GeoGebra macro, a Bezier curve, maybe at intervals, that approximates this track. Exporting this Bezier curve is a standard procedure. You can see an example in figure 12; the curve Γ is a complex curve, while all the others are simply circle arcs. It is in principle possible to find the parametric equations of Γ, but the use of GeoGebra capabilities makes everything extremely simple, without any calculation.

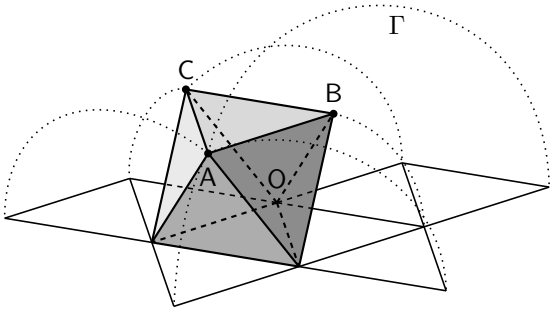

Figure 12: One of the different plane developments of the regular octahedron. The curves described by the vertices during development are highlighted

Once you have built the outline of a dodecahedron in the 3D window of GeoGebra, you can also experiment with interesting derived figures. An example is given in figure 13, where Leonardo's *Dodecahedron Planum Vacuum* is represented. Once more the calculation of the position of the vertices

of this figure is almost straightforward in GeoGebra, but it could be very difficult otherwise.

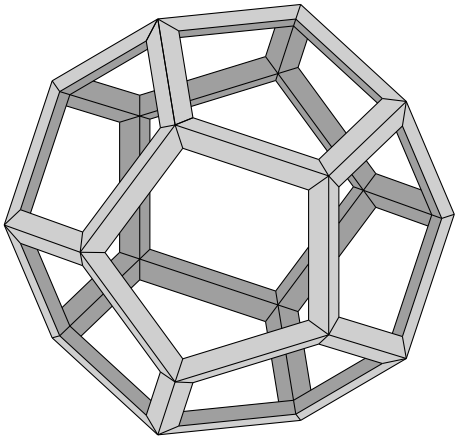

Figure 13: Dodecahedron Planum Vacuum, in Leonardo's style

6 The football

Once you have acquired familiarity with the Platonic solids, you can experience the expansion of the technique to other solids, i.e., the Archimedean solids. These can be obtained in various ways from the Platonics, for example by truncation starting from the vertices. In figure 14 we show the case of the icosahedron; given the Platonic solid, we consider, for each vertex, a sphere centered at the vertex itself and with variable radius. The intersection of this sphere with the sides of the polyhedron gives rise to regular pentagons and hexagons. The latter become regular when the radius of the sphere is exactly 1/3 of the side of the polyhedron and this situation corresponds to the truncated icosahedron. Using GeoGebra it is very easy again to document this process; simply project the truncation at the desired stage and then export it.

Figure 15 shows the final result. Nothing new is required in GeoGebra to obtain this last figure. It is exactly the same construction used for the previous figure 14, only with a different radius for the truncating spheres.

As is well known, the football is simply the projection of the truncated icosahedron on the circumscribed sphere. This can be achieved in different ways. In our opinion the simplest one is to project each side of the polyhedron onto the sphere by means of a parametric equation and then again to project the obtained arc in the 2D window. Following we describe the outline of this technique. Given a segment $\overline{\mathsf{AB}}$ with bounds $(x_\mathsf{A}, y_\mathsf{A}, z_\mathsf{A})$ and $(x_\mathsf{B}, y_\mathsf{B}, z_\mathsf{B})$, write the standard parametric equations of the segment

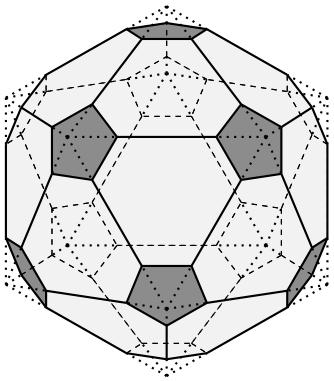

Figure 14: Outline of the truncation of the icosahedron starting from the vertices

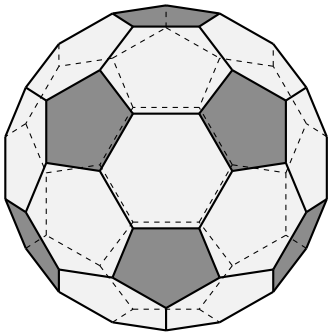

Figure 15: The truncated icosahedron

itself:

$$\mathsf{P}(t)\colon \begin{cases} f(t) = x_\mathsf{A} + (x_\mathsf{B} - x_\mathsf{A})t \\ g(t) = y_\mathsf{A} + (y_\mathsf{B} - y_\mathsf{A})t \\ h(t) = z_\mathsf{A} + (z_\mathsf{B} - z_\mathsf{A})t \end{cases}, \ 0 \le t \le 1.$$

Then find the norm of $\mathsf{P}(t)$:

$$||\mathsf{P}(t)|| = \sqrt{f^2(t) + g^2(t) + h^2(t)}.$$

The projection of the segment $\overline{\mathsf{AB}}$ on the unit sphere has the following parametric equations:

$$\mathsf{Q}(t) = \frac{\mathsf{P}(t)}{||\mathsf{P}(t)||}.$$

At this point there is nothing to do but use the already considered parallel projection form 3D to 2D to obtain a 2D curve. The final result for the football is shown in figure 16.

One last practical tip: the Ti*k*Z code of a figure like figure 16 is very long and complex (about 250 rows!) and it is useful to export it from GeoGebra one piece at a time, and not all together, especially if you need to paint the different parts in different ways (in our figure only black sphere pentagons and white sphere hexagons). It will be simpler to correctly fill the various parts of the figure, or to check if everything works correctly.

Figure 16: The football obtained by the projection of the truncated icosahedron on the circumscribed sphere

A simple but interesting application of this technique is shown in figure 17 where we have projected on the circumscribed sphere the regular tetrahedron. This figure solves an interesting problem: is it possible to cut an apple into four equivalent parts in an uncommon way?

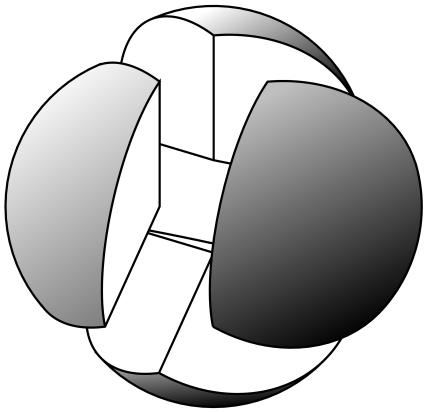

Figure 17: An apple cut in four parts in a non standard way

Before ending this "sport" section of our article we present a simple figure obtained from the truncated icosahedron: the molecule of the *Buckminsterfullerene*. In this case we have simply replaced the segments that make up the sides of the polyhedron by tubes and the vertices by shaded spheres. The following code for the tubes is taken from `https://tex.stackexchange.com`:

```
\newcommand{\Tube}[6][]%
{\colorlet{InColor}{#4}
 \colorlet{OutColor}{#5}
 \foreach \I in {1,...,#3}
 {\pgfmathsetlengthmacro{\h}{(\I-1)/#3*#2}
  \pgfmathsetlengthmacro{\r}{sqrt(pow(#2,2)
    -pow(\h,2))}
  \pgfmathsetmacro{\c}{(\I-0.5)/#3*100}
  \draw[InColor!\c!OutColor, line width=\r,#1]
    #6;
 }
}
```

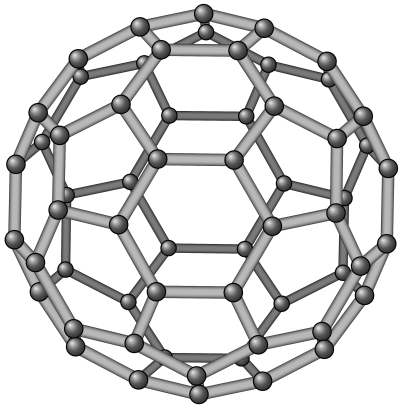

Figure 18: The molecule of Buckminsterfullerene

In a figure like this, in order to hide the invisible parts you need only to plot the rear parts first. As usual, you can easily locate them using the GeoGebra figure.

7 Conic and spherical sections

A very relevant problem for people interested in 3D graphics is the drawing of plots concerning conic sections. We only show some examples without extended details; the technique to be used is now familiar because, naturally, the involved curves are conics that GeoGebra can deal with using standard commands.

Figure 19 illustrates the two series of circular sections in an oblique cone; the sections parallel to the basis and the subcontrary sections, as considered by Apollonius. The complexity of this figure is due to mathematical calculations; you must find the correct angle for the plane that produces the subcontrary section, and the best way to do this is the original Apollonius description.

Figure 20 shows how to section a cone in order to obtain a hyperbola. The technique to obtain such a figure is simple in GeoGebra; after plotting the entire cone and the hyperbola on the cone you can hide, directly in GeoGebra, one of the two parts, leaving only the remaining one. After exporting the code you can shift, for instance, the right part using the following very standard code:

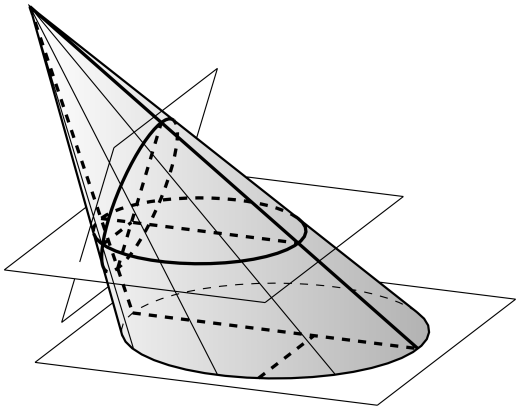

Figure 19: The two series of circular sections that can be obtained in an oblique cone

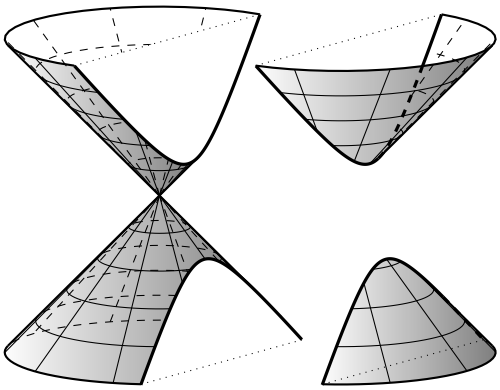

Figure 20: Section of a cone to obtain a hyperbola

```
\begin{scope}[xshift=2.4cm]
⟨code of the second part⟩
\end{scope}
```

The last figure of this section, figure 21, is somewhat more complicated, because GeoGebra can't handle directly (at least at the moment) the intersection between a cylinder and a sphere. Anyhow, the intersection curve is a Vivianis's window and the parametric equations can be found easily in all books of curves. The rest of the construction does not require special attention; there are only parts of a cylinder and of a sphere.

8 Some more advanced images

The technique based on exports from GeoGebra can also handle more complicated figures, but, naturally, a somewhat advanced knowledge of GeoGebra is required for this. In our opinion the effort is worth the candle because what you can obtain is very interesting. We give three images as examples.

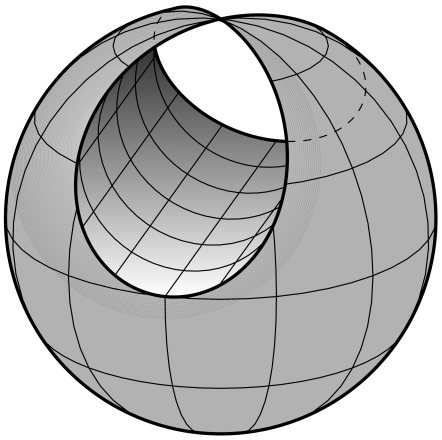

Figure 21: Intersection between a cylinder and a sphere

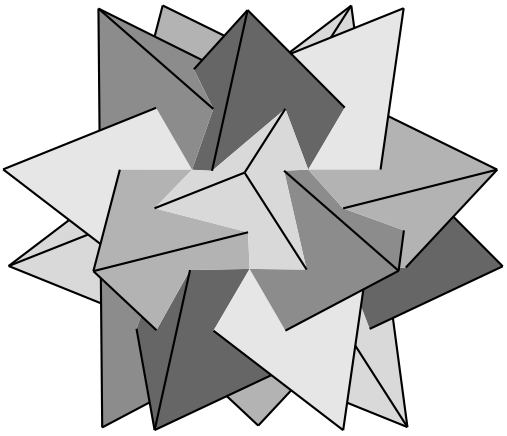

Figure 22: The compound of five tetrahedra

The first figure is the *compound of five tetrahedra*, which is one of the five regular polyhedral compounds. It can be constructed by arranging five tetrahedra inside a dodecahedron, having no common vertex. The correct construction of such a figure requires full attention, in particular to understand which are the actual sides and which instead are only fake sides, that must not be highlighted in the figure. Furthermore, in this case it is better to hide completely the non-visible part of the figure.

The second and third figures are reproductions of originals by Kepler, published in the 1596 in *Mysterium Cosmographicum*. They deal with a picture concerning the solar system as known in those times and consist of the five Platonic solids inscribed one into the other, while the inscribed/circumscribed spheres to each polyhedron contain the orbits of the six planets, earth included, with the sun at the center. As in the original by Kepler we propose both

the set of all the Platonic solids and a detail of the four interior spheres with the corresponding three polyhedra.

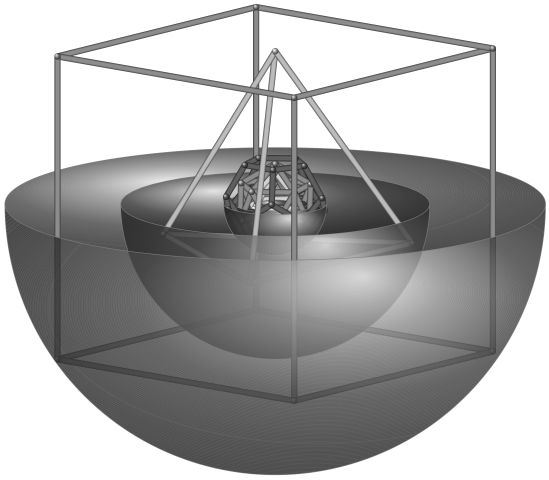

Figure 23: Reproduction of the solar system, as originally drawn by Kepler in 1596

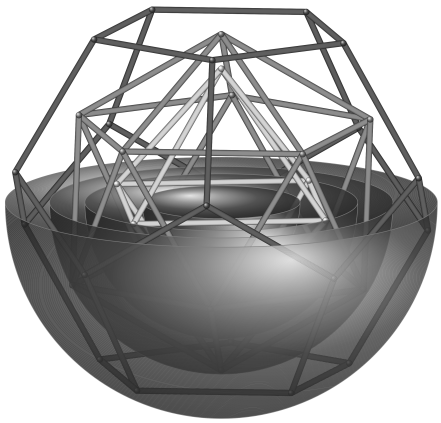

Figure 24: Reproduction of the solar system originally drawn by Kepler in 1596: detail of the central part

As with figure 18, one of the secrets for drawing correctly is to start from the back and to end by drawing the front parts of the figure.

9 Conclusion

We believe that the proposed technique can be advantageously used for the production of a large part of the geometric type figures required in a mathematical paper. This technique paired with pgf-plots or the corresponding packages for the PSTricks family allows the production of complex scientific books using LaTeX and without any external software.

As already mentioned, particularly when drawing complex figures, a somewhat deep knowledge of GeoGebra is required, but, in our experience, the learning curve of GeoGebra is much flatter than that of Ti*k*Z; the use of GeoGebra also offers the numerous advantages we have described in this article.

Naturally there is no rose without thorns and it is not possible to achieve the effects we have described without hard work and experimentation.

References

[1] Luciano Battaia. Grafica 3D con Geogebra e Ti*k*Z. *ArsTEXnica*, 22:50–63, 2016. guitex.org/home/images/ArsTeXnica/AT022/battaia.pdf.

[2] Claudio Beccari. The unknown picture environment. *ArsTEXnica*, 11:57–64, 2011. tug.org/TUGboat/tb33-1/tb103becc-picture.pdf.

[3] Agostino De Marco. Illustrazioni tridimensionali con Sketch/LaTeX/PSTricks/Ti*k*Z nella didattica della Dinamica del Volo. *ArsTEXnica*, 4:51–68, 2007. guitex.org/home/it/numero-4.

[4] Tomasz M. Trzeciak. texample.net/tikz/examples/map-projections/, 2008.

[5] Keith Wolcott. Three-dimensional graphics with PGF/Ti*k*Z. *TUGboat*, 33(1):102–113, 2012. tug.org/TUGboat/tb33-1/tb103wolcott.pdf.

◇ Luciano Battaia
Via Garibaldi 4
San Giorgio Richinvelda, PN 33095
Italy
luciano.battaia (at) unive dot it
http://www.batmath.it