Visualizing Polygonal Objects in 3D with Computer Algebra Systems: Waterman Polyhedra case study

Miroslaw Majewski
Zayed University
United Arab Emirates
mirek.majewski@mupad.com

Abstract

Polygonal objects are objects built out of multiple polygons. Examples of polygonal objects in 3D are convex and non-convex polyhedra and other various one- or two-sided shapes in 3D. In many cases modeling polygonal objects in 3D requires special algorithms, for example algorithms to calculate vertices and create polygons out of these vertices, algorithms to build a convex hull in case of convex objects, etc. Some of these tasks are very time consuming, especially while dealing with objects built out of hundreds of polygons. Therefore, algorithms to perform these tasks should be highly efficient.

In this paper I will describe construction of Waterman polyhedra and show how Waterman polyhedra can be created using Computer Algebra System MuPAD. I will briefly describe the algorithm that I developed to calculate Waterman polyhedra vertices. Finally, I will show a gallery of selected Waterman polyhedra.

1. Introduction

Waterman polyhedra were invented, around 1990, by Steve Waterman—a Canadian enthusiast of mathematics, physics and cartography. In fact, the idea of his polyhedra came up while solving a cartography problem.

Waterman polyhedra form a vast family of polyhedra. Some of them have a number of nice properties like multiple symmetries, or very interesting and regular shapes. Some others are just a bunch of faces formed out of irregular convex polygons. The most popular Waterman polyhedra are those with centers at the point $(0,0,0)$ and built out of hundreds of polygons. Such polyhedra resemble big spheres in 3D. In fact, the more faces has a Waterman polyhedron, the more it shape resembles the sphere circumscribed on it. Its volume and total area are close to those parameters of the circumscribed sphere.

With each point of 3D space, we can associate a family of Waterman polyhedra with different values of radii of the circumscribed spheres. Therefore, from a mathematical point of view, we can consider Waterman polyhedra as a 4D space $W(x,y,z,r)$, where x, y, z are coordinates of a point in 3D, and r is a positive number, usually $r \geq 1$.

Because of its complexity, and sometimes huge number of faces, Waterman polyhedra are a very interesting and challenging computational problem. In the last 10 years there were a few attempts to produce Waterman polyhedra using the Java programming language (Mark Newbold, [3]), Python (Kirby Urner, [4]), C++ and POV-Ray (Paul Bourke, [1]). Most of these implementations produce a family of Waterman polyhedra with the center at $(0,0,0)$ only. Mark Newbold developed a Java applet that is able to calculate and display seven families of Waterman polyhedra associated with some selected points in 3D. None of the mentioned implementations produces Waterman polyhedra for any point in 3D and any

positive radius.

In this paper we will show how Waterman polyhedra can be visualized in Computer Algebra System, MuPAD using its programming language and its powerful visualization tool—Virtual Camera. This is the first implementation producing Waterman polyhedra for any point in 3D and any positive radius. For detailed description of this method see [2].

The presented here method can be also implemented in Maple and Mathematica. In both cases a serious limitation will be lack of some graphical features that are available in MuPAD. There is no way to implement any method of creating Waterman polyhedra in Derive.

2. The concept of Waterman polyhedra and Waterman Polyhedra Space

We will start by examining the concept of Waterman polyhedra. Imagine that we place on a plane a bunch of spheres, all having the same radius, say $\frac{\sqrt{2}}{2}$, and organize them in such a way that the empty space between them is minimal, like in figure 1:

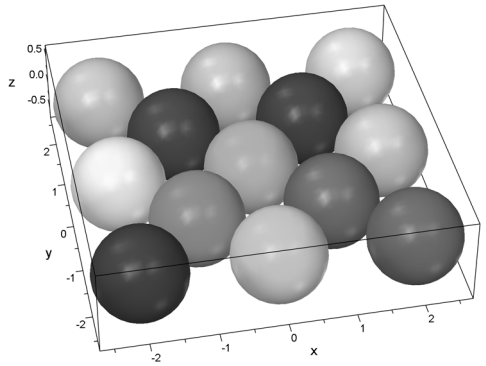

Figure 1 - A single layer of close-packed spheres

In figure 1, we used a finite number of spheres. However, in theory, we should consider an infinite, in each horizontal direction, layer of spheres. Now, on top of this layer we can put another layer of spheres in such a way that the new spheres will fall into holes between spheres of the first layer. Of course, we still assume that all spheres have the same radius. If we continue this process by adding more layers up and down of the first layer we will create a regular arrangement, often called the lattice arrangement, which is known in crystallography as Cubic Close Packing (CCP) or Face Centered Cubic arrangement (FCC). A finite example of CCP arrangement is shown in figure 2.

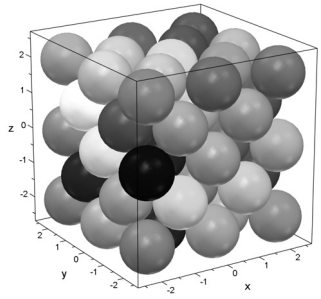

Figure 2 - Cubic Close Packing arrangement of spheres

Now, let us concentrate on centers of spheres of the CCP arrangement (figure 3). In this paper we will call

them CCP points. These are located on a grid of horizontal and vertical lines, used in crystallography to represent sodium chloride crystals. In this representation, chloride ions, shown as big dots in the figure 3, are arranged in a cubic close packing, while sodium ions fill the octahedral gaps between them. Each chloride ion has six nearest neighbors with octahedral geometry.

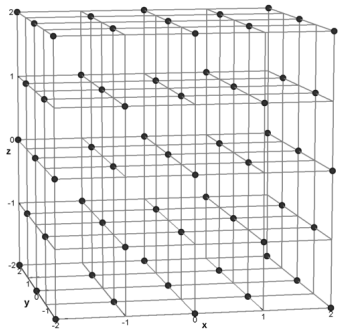

Figure 3 - Centers of spheres in CCP arrangement

In the next step, let us take a point $P(x,y,z)$ in 3D and develop a large sphere with radius R. In this case, some of our CCP points will fall inside of the large sphere, and some will not. If we remove all CCP points that are outside of the big sphere we will obtain a finite number of points that, in this paper, we will call Waterman points, associated with the point P and the radius R (see figure 4).

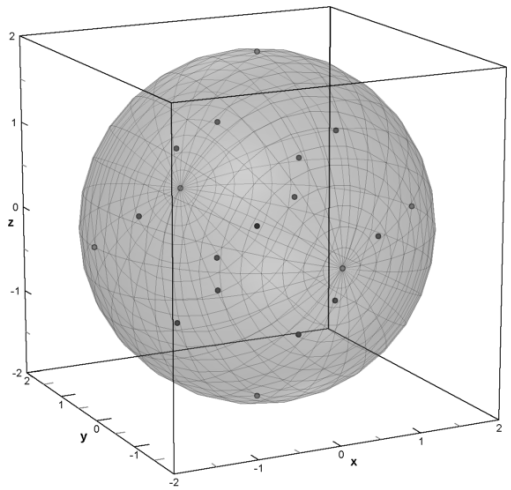

Figure 4 - A large sphere with Waterman points

In the final step we will take all the Waterman points, i.e. the points that lay inside of the big sphere, or on its surface, and use them to build a convex hull. This will be the Waterman polyhedron associated with the point $P(x,y,z)$ and the radius R. Figures 5a and 5b show Waterman polyhedra associated with the point $P(0,0,0)$ and radii equal to 2 and 5.5. We will denote a Waterman polyhedron with the center at $P(x,y,z)$ and radius R as $W((x,y,z),R)$.

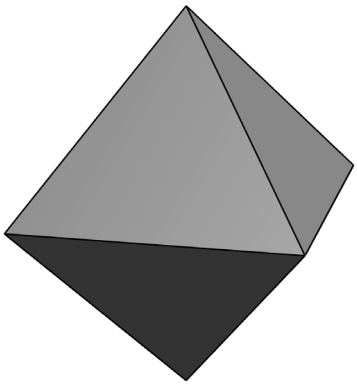

Fig. 5a – $W((0,0,0),2)$

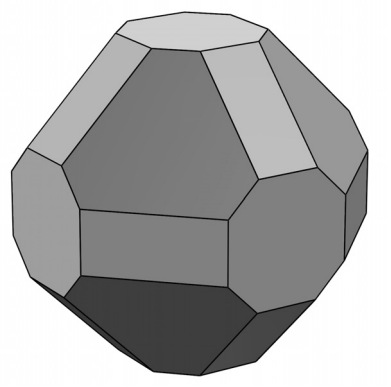

Fig. 5b – $W((0,0,0),5.5)$

In our examples, for the CCP arrangement, we used spheres with a radius equal to $\frac{\sqrt{2}}{2}$. Therefore, we may easily notice that coordinates of centers of spheres forming the CCP lattice are integer numbers m, n, and p, such that $m+n+p$ is an even number. A simple consequence of the CCP lattice structure, and how Waterman polyhedra are created, is that

$$W((x+2k,y,z),R) = W((x,y+2k,z),R) = W((x,y,z+2k),R) = W((x,y,z),R) \qquad (*)$$

k is an integer number. Therefore, we can restrict investigations of Waterman polyhedra to the points (x,y,z) in the cube $[0,2)\times[0,2)\times[0,2)$ and any positive radius R.

On his web site, Mark Newbold investigated seven sequences of Waterman polyhedra (see [3]). Each sequence was created by starting from a fixed point in 3D as the center of a Waterman polyhedron and changing radius of the big sphere. Although Mark Newbold used slightly different points as centers of his Waterman polyhedra sequences than those shown in the figure 6, according to our observation above (*) we can use points that are equivalent to Newbold's points and are located in the cube $[0,2)\times[0,2)\times[0,2)$, or even in $[0,1)\times[0,1)\times[0,1)$. These are shown below, in figure 6.

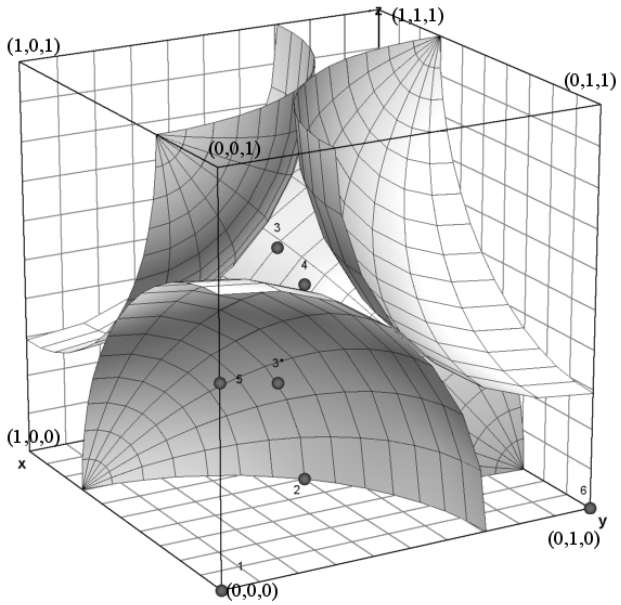

Figure 6 - Seven origins of Waterman polyhedra sequences by Mark Newbold

Table 1 presents detailed information about these seven points.

Origin	*Origin location*	*Description*
1	$(0,0,0)$	*Center of a sphere*
2	$(1/2,1/2,0)$	*Point where two spheres touch, e.g.* $((0,0,0)+(1,1,0))/2$
3	$(1/3,1/3,2/3)$	*Center of the space between three spheres, e.g.* $((0,0,0)+(0,1,1)+(1,0,1))/3 = (1/3,1/3,2/3)$
*3**	$(1/3,1/3,1/3)$	*Center of the space between three empty nodes of the CCP lattice, e.g.* $((1,0,0)+(0,1,0)+(0,0,1))/3 = (1/3,1/3,1/3)$
4	$(1/2,1/2,1/2)$	*Center of the space between four spheres or for empty nodes of the CCP lattice, e.g.* $((0,0,0)+(1,0,1)+(0,1,1)+(1,1,0))/4 = (1/2,1/2,1/2)$
5	$(0,0,1/2)$	*Point half way between sphere center and empty node of the CCP lattice, e.g.* $((0,0,0)+(0,0,1))/2 = (0,0,1/2)$
6	$(1,0,0)$	*Empty node of the CCP lattice*

Table 1 – Origins of Waterman polyhedra sequences by Mark Newbold

Finally, in order to create his sequences of Waterman polyhedra Mark Newbold used specific radii. The main reason for this was that, quite often, even a very small change of the radius creates a new polyhedron. Table 2 presents the radii formula for each sequence of Waterman polyhedra by Mark Newbold. Note that the radii formulae in the table 2 were modified in such a way that for each sequence the parameter n in the calculations of Waterman polyhedra starts from 1 and is exactly the same as the number of the polyhedron in the sequence. In Newbold's original work, the parameter n in the calculations is not always the same as the number of the polyhedron in its sequence. Note also that the name of the origin is not always the same as the number of the sequence .

Sequence #	*Origin name*	*Radius formula for* $n = 1,2,\ldots$
1	*Origin 1*	$R = \sqrt{2n}$
2	*Origin 6*	$R = \sqrt{1+2(n-1)}$
3	*Origin 4*	$R = \frac{\sqrt{3+8(n-1)}}{2}$
4	*Origin 2*	$R = \frac{\sqrt{2+4n}}{2}$
5	*Origin 5*	$R = \frac{\sqrt{1+4n}}{2}$
6	*Origin 3*	$R = \frac{\sqrt{6(n+1)}}{3}$
7	*Origin 3**	$R = \frac{\sqrt{3+6n}}{3}$

Table 2 – Radii formulae for Mark Newbold's sequences of Waterman polyhedra

In the further pages of this paper, we will use notation $w_{k,n}$ to represent the n-th Waterman polyhedron from the sequence k, where $k = 1..7$. For example $w_{3,10}$ is the tenth Waterman polyhedron from sequence 3.

Figure 7 presents two examples of Waterman polyhedra from Newbold's sequences, $w_{3,45}$ and $w_{7,77}$. In one of the later chapters we will spend more time exploring Waterman polyhedra. For this purpose we will use the Waterman library created by the author in MuPAD.

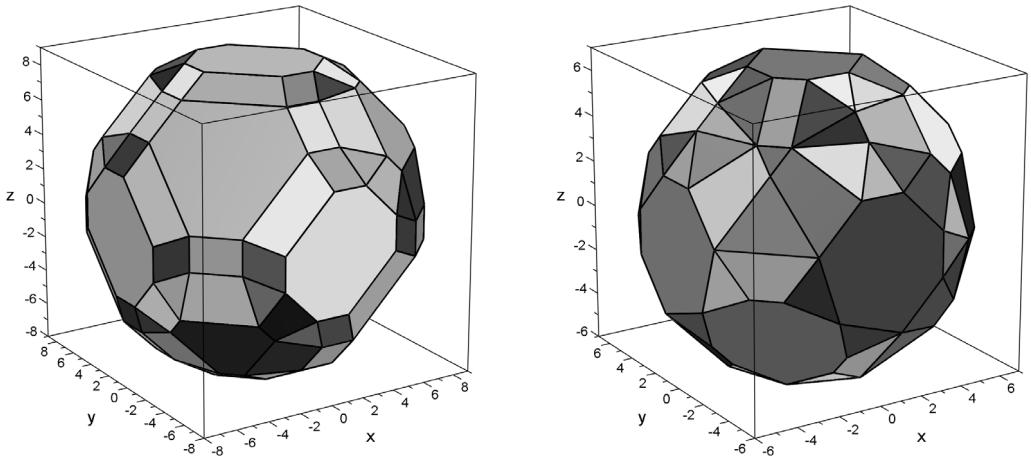

Figure 7 - Waterman polyhedra $w_{3,45}$ and $w_{7,77}$

3. Waterman library installation and configuration

The Waterman library in MuPAD was created as a response to Steve Waterman's question if MuPAD can be effectively used to visualize large objects with thousands of polygons. The first attempt was to create a procedure that will produce only polyhedra $w_{1,n}$, i.e. Waterman polyhedra from the first sequence, the one with the origin at $(0,0,0)$. The same code with significant changes was then used to produce the polyhedra $w_{k,n}$, where $k = 1..7$, as well as the procedure to explore the space of Waterman polyhedra. Finally, some additional tools for visualizing CCP spheres and the CCP grid were added.

At the time of writing this text, June 2006, the Waterman library is not a part of MuPAD original setup. However, tools for exploring Waterman polyhedra, based on this library, will be included in the next version of MuPAD.

The installation of the library requires two files, `waterman.mb` and `qconvex.exe`. The QConvex program is part of the Qhull library developed at the National Science and Technology Research Center for Computation and Visualization of Geometric Structures, The Geometry Center, University of Minnesota. The complete Qhull library can be downloaded from `http://www.qhull.org`. The `waterman.mb` library only uses `qconvex.exe` from the Qhull library.

Both files should be copied into the folder where in user configuration of MuPAD the variable `READPATH` points up. For example, in my configuration of MuPAD, all my works are in the folder `d:\documents\mupad`. Therefore in MuPAD I have to execute the command:

```
READPATH := "D:\\Documents\\MuPAD\\":
```

Finally, in order to have access to `qconvex.exe` from MuPAD, we need also an additional variable:

```
QCONVEX_PATH := "D:\\Documents\\MuPAD\\":
```

If both files were copied to the folder mentioned above, we can start a new MuPAD notebook with these commands:

```
READPATH := "D:\\Documents\\MuPAD\\":
read("waterman.mb")
```

```
QCONVEX_PATH := "D:\\Documents\\MuPAD\\":
```

Now we can execute any procedure from this library. The Waterman library contains the following procedures:

`CCPoints(xrange, yrange, zrange)` —*procedure to calculate CCP points, obtained output is a plot object that can be displayed on the screen using the* `plot` *command. Here is an example,*

```
plot(CCPoints(-2..2,-2..2,-2..2), Scaling=Constrained)
```

The result was shown in figure 3.

`CCSpheres(xrange, yrange, zrange)` —*procedure to calculate CCP spheres arrangement for a parallelepiped defined by* `xrange`*,* `yrange`*,* `zrange`*. Note, the output will contain all CCP spheres that have centers inside of the parallelepiped.*

`WatermanData([x,y,z],r)` —*procedure to calculate data for all procedures displaying Waterman polyhedra.*

`WatermanSpheres([x,y,z],r)` —*procedure to calculate all CCP spheres, with centers on, or inside a sphere with the given radius* `r` *and the center* `[x,y,z]`*. The obtained result is a plot object that can be displayed using the* `plot` *command.*

`WatermanPoints([x,y,z],r)` —*procedure to calculate the centers (CCP points) of all spheres produced by the* `WatermanSpheres` *procedure. The obtained result is a plot object that can be displayed using the* `plot` *command.*

`WatermanPoly(sequence,number)` —*procedure to obtain Waterman polyhedra from Mark Newbold's sequences. The obtained result is a plot object that can be displayed using the* `plot` *command.*

`WatermanExplore([x,y,z],Radius)` —*procedure that can be used to produce a Waterman polyhedron for any point of the 3D space as a center of the polyhedron, and for any positive radius. If such a polyhedron does not exist, the output is the point* $P(x,y,z)$*. In each case, the obtained result is a plot object that can be displayed with the* `plot` *command.*

Most of the procedures in the Waterman library produce objects with completely random colors. The advantage of this approach is that each face of a polyhedron gets an individual character. Therefore, while discussing properties of a polyhedron we are able to refer to the individual faces of it by using their colors. For example, we can say—the blue pentagon on the top of the polyhedron is ... etc. There are also serious disadvantages to such approach, however. For instance, animation of a polyhedron $W((0,0,a),3)$, where a is the animation parameter, will produce a series of blinking images.

I believe coloring polyhedra in MuPAD is another interesting topic that requires more attention.

Another natural question would be to investigate symmetries in Waterman polyhedra and show the one with the largest number of symmetries of a particular type. For example, the polyhedron obtained for $P(0,0,0)$ and $R = 21$ shows a number of rotational and reflexive symmetries,

For very large radii, we obtain polyhedra that look like a big jewels or mysterious crystals. In appendix we show the polyhedron $w_{2,1000}$ obtained in MuPAD. Analyzing its structure could be quite a challenging job. We will leave further explorations of Waterman polyhedra for another opportunity. I suggest to MuPAD users to install the Waterman library and continue these interesting investigations on their own.

4. Implementation in MuPAD

Implementing Waterman polyhedra in MuPAD created three interesting computational problems: how to obtain the set of vertices of a given Waterman polyhedron in the most efficient way, given that most known algorithms were too slow to implement them in MuPAD; how to create the polyhedron structure having its vertices; and finally, how to create such a polyhedron in MuPAD using the means available in OpenGL.

As we said already, there are known algorithms to produce points in 3D that can be used to create a Waterman polyhedron. However, most of them are too slow to be implemented in MuPAD. For example, the algorithm used in [1] to create Waterman spheres, after transforming it to MuPAD, turned into code with runtime complexity equal to

$$(2(2n+1)+1)^3 = 64n^3 + 144n^2 + 108n + 27$$

Such an algorithm will work considerably well in binary, i.e. compiled, programs. However, programs in MuPAD, like in any other Computer Algebra System, are interpreted and therefore much slower. So, we needed a fast and very efficient algorithm. For this purpose, an algorithm was created to cut a sphere with a given radius R into slices and choose from them only those points that have coordinates $x + y + z$ divisible by 2. With some improvements done by Oliver Kluge from SciFace Software GmbH., the algorithm produces results in a considerably shorter time. For instance, in MuPAD in order to produce $w_{1,100}$, using the algorithm from [1], we need 1,096,586 ms, which amounts to about 18 min. On the other hand, the optimized algorithm produces $w_{1,100}$ in 2,834 ms, which is about 3 sec.

Producing a set of CCP points was the very first step towards producing a Waterman polyhedron. In the next step, we need to filter out those points that do not lie on the surface of the polyhedron, and use the remaining points to create a convex hull, i.e. the smallest polyhedron containing our points. Instead of implementing a very complex algorithm to produce a convex hull, which would certainly be much slower in MuPAD than in binary code, I decided to use a ready-made program, `qconvex.exe`, written in C++. As I mentioned before, the QConvex program is a part of the Qhull library developed at the National Science and Technology Research Center for Computation and Visualization of Geometric Structures, The Geometry Center, University of Minnesota. The complete Qhull library can be downloaded from `http://www.qhull.org`.

The simplest way to use QConvex on data produced by the `WatermanData` procedure is to convert these data to a format acceptable by QConvex, save the data into a text file, apply QConvex on this file and then read the produced results back into MuPAD.

The next step would be to use this data to produce the faces of a polyhedron. For this purpose I used the `SurfaceSet` plot class from MuPAD plot library.

5. Literature

1. *Bourke P., http://astronomy.swin.edu.au/~pbourke/geometry/waterman/*
2. *Majewski M., Visualizing Waterman Polyhedra with MuPAD, at http://www.mupad.com/mathpad/2006/majewski/*
3. *Newbold M., http://dogfeathers.com/java/ccppoly.html*
4. *Urner K., http://www.4dsolutions.net/ocn/*
5. *Waterman S., Waterman Polyhedra, Symmetry: Culture and Science, Vol. 13, No.1, 2002*

Appendix: Gallery of selected Waterman polyhedra

In this section we will show examples some of the most interesting Waterman polyhedra. All examples were obtained using the `waterman.mb` library. In a few cases the code was modified to obtain a specific coloring effect.

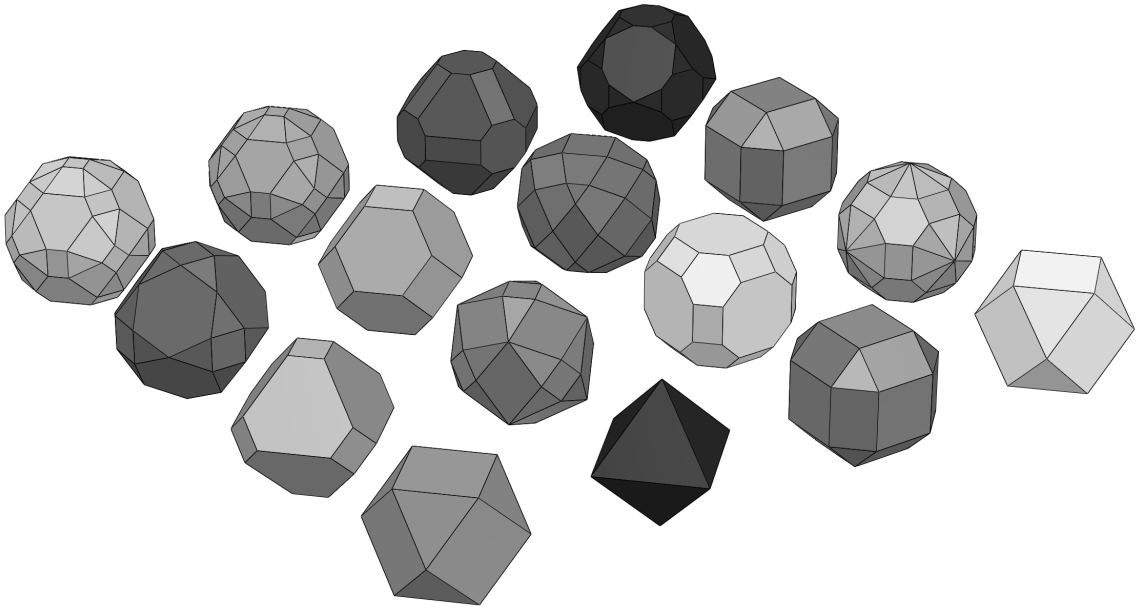

Figure 8 - Waterman polyhedra $w_{1,1}$ (left bottom corner), $w_{1,2}$, ..., $w_{1,16}$ (top right corner)

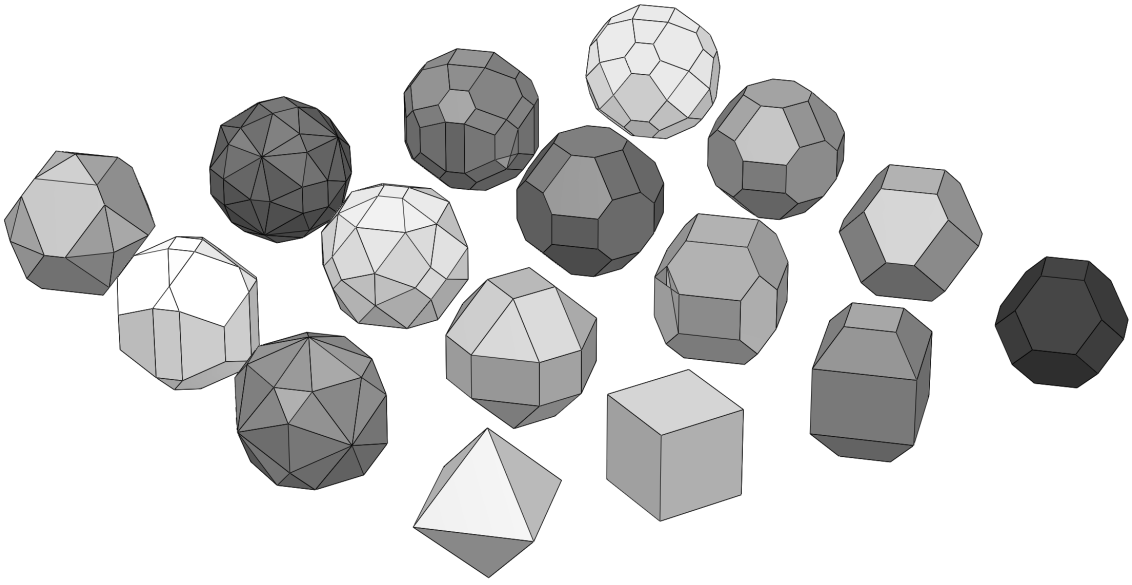

Figure 10 - - Waterman polyhedra $w_{2,1}$ (left bottom corner), $w_{2,2}$, ..., $w_{2,16}$ (top right corner)

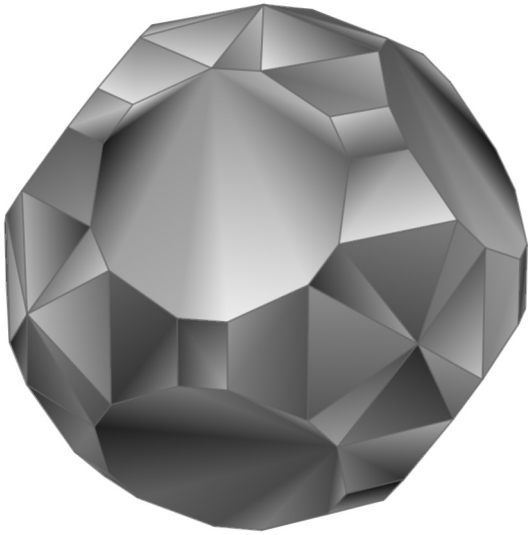

Figure 9 - Rainbow coloring scheme applied to Waterman polyhedron $W((0,0,0),8)$

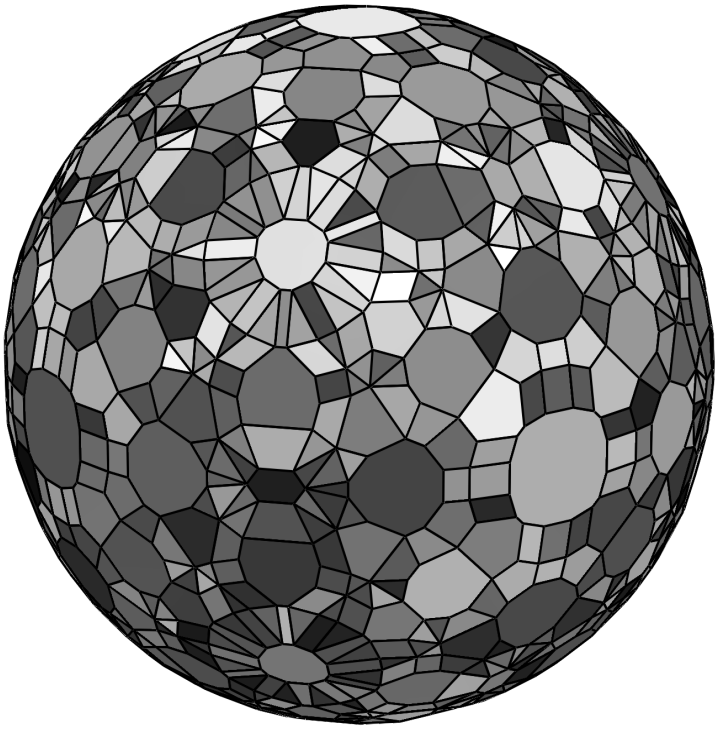

Figure 11 - Waterman polyhedron $w_{2,1000}$