"3d-OpenSHuFFle" OpenEuphoria program, version 1.1
"Shell-Hull File Format 1.1" ⇒ ".shuf" extension
“Section Arrangement Sequence (& File) 1.1” ⇒ ".shuf" extension

OpenShuffle is a program that converts a ".shuf" file descriptor into an Openscad ".scad" file. This file contains a number of Openscad Polyhedron, that all together compose the object/shell/hull described in the ".shuf" file. The ".shuf" file descriptor is composed of a "Main Sequence" (#start...#end) and of a "Section Arrangement Sequence" (#rotstart...#rotend). The arrangement sequence can be on a separate file, too, with the same extension of the main sequence. This ".shuf" file, if used, can rotate/scale/move the sections to simplify the design process of very complex objects.

Q.: Why a new program (in a different language, too), if exists OpenSCAD code that do something similar?

A.: Because OpenSCAD, although being a consistent graphic language, isn't so flexible to do a classic GUI with buttons, lists and edit-text fields, that sometimes could be useful. Moreover, some mathematical operations (and correlated data-managing) that are tricky under OpenSCAD, become very simple in Euphoria.

Q.: Why the need of the "*.shuf" file format?

A.: The reason to present this new file format derive from the difficult to use the minkowski() function in Openscad. If you want to make a thin shell, the first idea is: I make a solid object, with minkowski() I swell it up a little, then with difference() I can subtract the second from the first. The drawback is the time needed by OpenSCAD to perform this few operations: it increases very fast with the complexity of the object, so you will abandon this way very soon.
If you have got a "*.shuf" file of the exterior of a thin shell object, it's not so difficult to make the interior simply using a text editor. I hope to automate this process soon.

Q.: Why to use older versions of Euphoria and Win32Lib than the last ones?

A.: OpenEuphoria 3.1.1 is a very robust development environment, with minor bugs. It's simple to develop code that don't make surprises. Win32Lib is a worthy counterpart of OpenEuphoria; the only serious flaw that occurs when unfreezing the PC with an application opened (on Win XP-Win 7): sometimes you get a lot of "crazy" dialog boxes...

Not every match Euphoria-Win32Lib gets success, but Eu. 3.1.1 with Win32Lib 0.45r do: it's enough for my purposes.

".shuf" file: Main Sequence

Basically, the ".shuf" file is a sequence of m "sections", composed of n triplets of coordinates.
In other word:

```
#start
X11 Y11 Z11
X12 Y12 Z12
...
...
X1n Y1n Z1n
#2
X21 Y21 Z21
X22 Y22 Z22
...
...
X2n Y2n Z2n
.
.
#m
Xm1 Ym1 Zm1
Xm2 Ym2 Zm2
...
...
Xmn Ymn Zmn
#end
```

Numeric entries can be arranged as you prefer (also into a single line), however 3 coordinates per line is simpler to write and check.
Every section, composed of a list of points (the number of points "n" must be a constant in the sections of a single file), is the contour of a geometric section of the shell/hull that you are trying to represent. For **section = 1** ("#start" section) you describe the first face of the shell, for **section = m** the last face: **only these should be plane sections**, although it's not mandatory.
Every other section describes a face of tubular envelopes that connects section X points and section X-1 points. The envelope is composed by triangles. Obviously, the sections are very simpler to design if they lie on a plane parallel to the XY, XZ or YZ planes. In the following the sections are parallel to the XZ plane:

```
#start
X11 Y1 Z11
X12 Y1 Z12
...
...
X1n Y1 Z1n
#2
X21 Y2 Z21
```

X22 Y2 Z22
...
...

It's important: **Y1...Yn should be an increasing (or decreasing) sequence**: backward (or forward) jumps can lead to impossible/wrong objects.
Beware: caps matter! **All uppercase or lowercase**, not a mix. The X11....Zmn are numeric values: algebraic expressions not admitted. The numbers after the section separator "#" can be omitted, or substituted with a label, e.g.:

#22_neck

Beware: **no spaces admitted** between "22" and "neck"!!! Only an alphanumeric symbol.
After the "#end" line (sections terminator) and before the "#start" line you can write everything (credits, copyrights etc.).
This is the basic structure of a ".shuf" file.

There are 6 more options:

<1> incremental operator "@": if this symbol come first of a coordinate (no spaces admitted), it's intended as an increment (or decrement if negative) from the corresponding element of the section-1 list. It's a very powerful option, that can accelerate by much the design process. You can mix normal and incremental coordinates as you prefer.

<2><3> global translation and rotation directives: global rotation (#rotate item) will be executed first, then global translation (#translate item), with no regard of their relative position in the input "*.shuf" file. However, it's a good idea to place them in the very first lines of the file, before "#start". Only the first occurrence of each one is processed. The order of execution is: first X axis, then Y, then Z.
They must be in the same file of the Main Sequence.

#rotate 0 12.0 43.5
#translate 25 27 -18.4
#start
...

<4><5> elliptical sections: "+circlexy", "+circlexz". The number of points must be the same of the other sections. Examples:
#start
+circlexy
43.1 55.6 28.0 (X, Y, Z of the center)
42.3 44.6 24 (semi-axis X, semi-axis Y, number of points)

#2 +circlexz 40.1 54.0 26.4 42.3 44.6 24
...
#end

N.B.: comments into parenthesis are ignored by the program.

Incremental symbol "@" not admitted in +circle.. arguments.

<6> NACA 4 digit airfoil section (number of points limited to 48). Example:
#20
+NACA4210 400 0 2800 0 48 (type of profile, chord, first point x, f.p.y, f.p.z, n. of points)
#21
...

The "First Point" is the leading edge of the airfoil. Beware: in the transformations of a section you must declare FP before the relative line in the "rotfile", then move into the section of a percentage of the chord. Otherwise, the trans. will be applied to the "Central Point" (mean point of the section), that's located at about 34% of chord from the leading edge (it vary slightly with the camber). If null camber, it's exactly at 0.3385 of chord.

Design style tips: if you are working with a lot of incremental coord. (ex.: "@48.3"), could be a good idea to put sometimes a "neck", i.e. a whole section composed of absolute coordinates. This let you to control very carefully the shape in crucial points, avoiding excessive swelling or shrinking. This section is simple to arrange: apply <Save absolute SHuFFle> to the "*.shuf" file (Rotfile Off/darker b.ground), then you can pick only the section you need and replace it in the original file.

".shuf" file: Section Arrangement Sequence (or "Rotfile")

The Section Arrangement Sequence defines the relative movements between sections and scaling actions. It can lie together (in the same file of) the Main Sequence, or in a separated file (with .shuf extension as well).

It supports every type of rotation: around an axis parallel to X, then parallel to Y, then parallel to Z.
There are 2 types of Rotfile: the simple one and the extended. In the next example of the simple one, the sections are parallel to XZ plane:

#rotstart
CP 8 0 0 0 0 0 ([CP/FP switch] -- fixed point offset x, y, z -- rel. rotations rx, ry, rz)
#1
8 0 0 0 0 2 (fixed point offset x, y, z -- rel. rotations rx, ry, rz)
#2
FP 9 0 0 0 0 4.5 ([CP/FP switch] -- fixed point offset x, y, z -- rel. rotations rx, ry, rz)
#3
10 0 0 0 0 6.0 (fixed point offset x, y, z -- rel. rotations rx, ry, rz)
...
#rotend

The type of fixed point (of rotation) can be selected between "**Central Point**" (CP) and "**First Point**" (FP). Putting a "CP" or "FP" flag at the start of a section arrangement line, it changes the way used by the program to evaluate the new section position. This behaviour is maintained from that section until a new flag is found, so these flags don't need to be repeated on any line: after the

first flag, located in the first line after "#rotstart", every line will be treated in the same manner. If not specified otherwise, default is "CP".

Normally the x, y, z coord. of the "**Fixed Point Offset**" are all 0's, but in some cases could be useful to have them different than zero. The rotations are applied in this way: first rx, then ry, then rz.

NOTE: if Fixed Point Offset is {0, 0, 0} and "CP" flag (or no flag) is specified, the Fixed Point and the "Central Point of Section" (CPS = sum of coordinates X, Y, Z divided by the number of points) are the same. If Fixed Point Offset is {10, 24, 0}, then Fixed Point will be located in CPS + {10, 24, 0}. If Fixed Point Offset is {0, 0, 0} and "FP" flag is specified, the Fixed Point will be the First Point of the section. Obviously, The CP of an elliptic section will be the centre of the ellipse.

The numbers after the "#" symbol should match with the similar numbers in the ".shuf" file (although it's not mandatory). The number of arrangement lines must be equal or bigger than the sections. Symbol "@" not admitted.

NOTE: the #rotstart-#rotend sequence can be included in a ".shuf" file containing a #start-#end sequence, tipically at the end. "Rotfile" must be choosen anyway, selecting the same "*.shuf" file for arrangements too. The function of the <Rotfile On/Off> button is the same.

In the extended version you can add one or more parameters, until a max of 6 (totally 12):

```
#rotstart
 8 0 0  0 0 0    1.0 1.0 1.0  00 00 00    (fixed point offset, relative rotation, scale factors, relative move)
#1
 8 0 0  0 0 2    1.1 1.0 1.0  00 00 00
#2
 9 0 0  0 0 4.5  1.2 1.0 1.2  00 00 01
#3
10 0 0  0 0 6  1.3 1.0 1.3  00 00 01
#rotend
```

Beware: the representation of every single entry is slightly different (0, 1, -12 etc. / 1.0, 1.1 etc. / 00, 013, -07) only for visual/mnemonic purposes. Entries can be always provided in the xxx.yyy notation.
The rotfile line, now, is composed of a minimum of 6 to a maximum of 12 elements, where the first 6 are unchanged. The last 6 specify the "scale parameters" on the 3 axis ("1.0 1.0 1.0" doesn't change anything) and the "move parameters" ("00 00 00" doesn't move).
After the first 6 values, you can stop the line at any moment: "1 1 1 0 0 0" sequence is the default for missing elements.
"Scale" is a vector of scaling factors, one for any axis. Nevertheless, for sections on xz plane don't make sense Y-scale values different than 1; the same for sections on xy plane and Z-scale.
"R. Move" (relative move) vector describe a translation of a given section related to the former one; sometimes could be very useful, in most cases this entries can be omitted.

REMEMBER THAT R. ROTATION AND R. MOVE OF SECTION "N" ARE RELATIVE TO THE SECTION "N-1", so sections N+1, N+2... are affected too! If you apply r. rot. and r. move to the very first section, the entire object is rotated/moved.

Shuffle User Interface

Buttons on the User Interface:

[1] "SHuFFle to Empty Polyhedron" (SEP): it can represent grafically every single shell you made, no matter their complexity; but the OpenSCad capability to manage these objects, as well as imported “.stl” files, could be reduced (boolean operations, etc.) and/or slowed down. When mixing +circle sections and classic sections (i.e. list of boundary points) be careful: if the starting point and/or the spatial direction of the points list don't correspond, the result will be embarrassing. This problem doesn't affect the "SHuFFle to Solid Hull" function, instead.
- OpenSCad can do SEP objects rendering faster than the SSH ones.
- The objects created are solid polyhedrons, despite the name, whereas SSH objects are composed of hulls (OpenSCad “hull()” function applied to polyhedrons).

[2] "Save Absolute SHuFFle": useful to extract sections of “.shuf” file (translated and rotated as requested) in absolute coordinates. Not affected by the RotFile button, but affected by #rotate and #translate directives.

[3] "SHuFFle to Solid Hull" (SSH): can represents correctly only .shuf files vith convex boundary sections. If some sections aren't so, you can try to divide the file into two (or more) convex sections files. The resulting “.scad” objects are easier to manage with boolean operations.

[4] "RotFile on/off": ToggleButton that switch from linear creation style to modified style.

[5] "SHuFFle to Support Array": generates a .scad file that contains a Smart Support Array.
This is the result of a geometric intersection() operation between the "empty_space()" module (e.g. the input "empty_space.shuf" file transformed in an OpenSCAD solid object) and an array of 20x20 mm universal supports. This empty_space() is defined with the geometry generated by the "SHuFFle to Solid Hull" function.
The "empty_space()" module is located in the first lines of the output file. That's the OpenSCAD version of the provided ".shuf" file; this file should be in the "cream-puff" style: the bottom flat and parallel to X-Y plane, external walls nearly vertical. This “.shuf” file represents the inner empty space of a thin shell, generally.
This function is presented in a very early version, so you don't be surprised of strange results. In the "*_SA.scad" output file there's a numerical array filled with the heights of the columns that forms the support of the object to be printed: putting a zero each support can be disabled. The heights can be arranged manually to obtain an optimum.
The goal is to minimize the amount of support material needed in the print of thin shells nearly parallel to the bed. The material saving can result up to 60-70%. Obviously, if inclination of top is under about 52° from vertical, only supports for model stability are needed.
Important: the shuf file sections must be arranged to develop in the "y" axis.
After the last developments in slicing techniques, this function is very less interesting; nevertheless, it could be useful for some unusual projects.