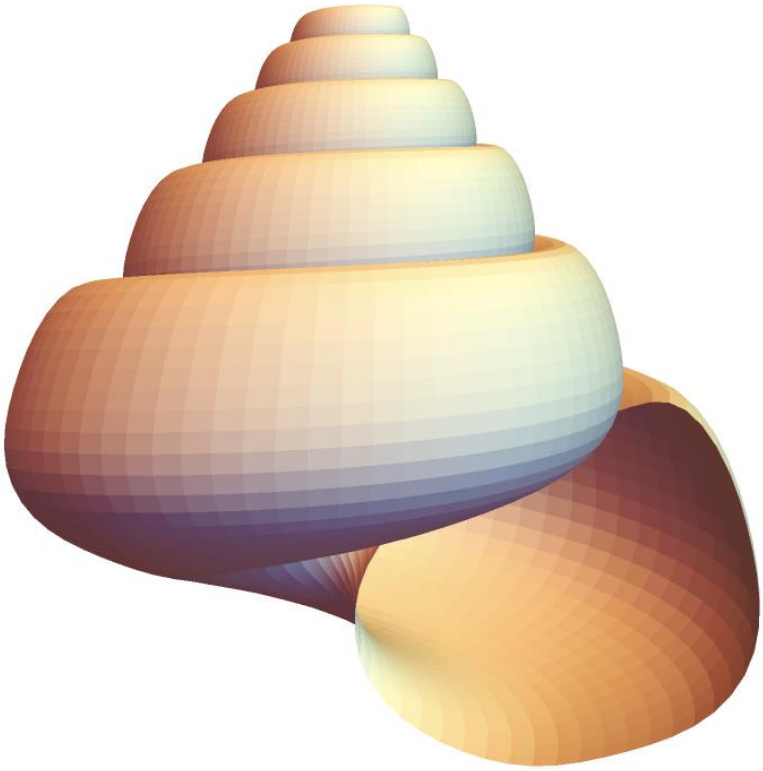
Snails for Mathematica

User's Guide

Version 1.1, October 2025



P. David Polly

https://pollylab.org/

pdpolly@pollylab.org

Cite as:

Polly, P.D. 2025. Snails for Mathematica. Version 1.1. https://github.com/pdpolly/Snails-For-Mathematica

Table of Contents

Installing the package

1. Download the latest version of the package at https://pollylab.indiana.edu/software/index.html (right click on link to save as file)
2. Open the file in *Mathematica*
3. Under the File menu, choose “Install”
4. Under Type of Item choose “Package”, under source choose the file you just saved, under Install Name choose a short name for the package (e.g., “Snails”)
5. Once installed, enter the command “<<Snails`” to use the functions.
6. Use the function *SnailsVersion[]* to determine which version you have installed.

Using Mathematica

Mathematica has a unique interface that takes a while to get used to. You open to a blank page, like a word processor, where you can type anything you want. Most of the time you will type commands that do things with your data: connect to databases, plot graphs, carry out calculations. Unlike other statistical or mathematical programs, the commands you type and the output you get remain on the page, which gives you a record of what you've done step-by-step. To help organize your work, you can add headers, format boxes, etc. with the Format Menu.

Cells are an important organizing feature of *Mathematica*. Note the "cell" markers on the right margin. Each cell is bounded by a bracket and commands within a cell are executed together. You can open and close cells by double clicking the bracket. This can be useful if you have lots of stuff in a notebook… you can give a section a heading, which causes cells in that section to be grouped, after which you can close the section by double clicking.

Shift + Enter causes a cell to be executed. Pressing the enter key creates a new line, just like in a word processor, but when you want to execute a command you typed, you type SHIFT+ENTER somewhere in the cell and all commands in the cell are executed.

***Mathematica* Commands**. Mathematic is designed to be as easy to learn as possible so that you can concentrate on working instead of the program. Almost all commands are English words written out in full with capitals at the beginning of words and brackets [] at the end of the command. For example, the command to calculate an average of a set of numbers is *Mean[]*, the command to do a principal components analysis is *PrincipalComponents[]*, and the command to take the logarithm of a number is *Log[]*.

Formatting Output. Mathematica is clever about how it provides output and it tries to keep the results as accurate as possible. For example, if you calculate the average of the following numbers

```
Mean[ {1, 5, 10, 3, 20, 40} ]
```

the answer extends to many decimal places, so Mathematica reports it more precisely as a fraction: 79/6. You may want an ordinary number, however, and you can force Mathematica to format its output the way you want:

```
Mean[ {1, 5, 10, 3, 20, 40} ] // N
```

This command now gives you 13.1667. Another useful formatting function is **//MatrixForm**, which causes a table to be displayed neatly in columns instead of wrapping around the page.

Graphics. *Mathematica* is good at graphics. You can either use simple functions like *ListPlot[]* to create a generic graph, or you can experiment with *Graphics[]* to create a completely customized graphic.

RaupCoil3D[*shape, W, T, D, TurnNum (,True)*]

Created 16 March 2016
Updated 12 July 2022 to add coordinate option.

This function generates a three-dimensional rendered shell using the functions developed by David Raup (1966, 1967).

Arguments:

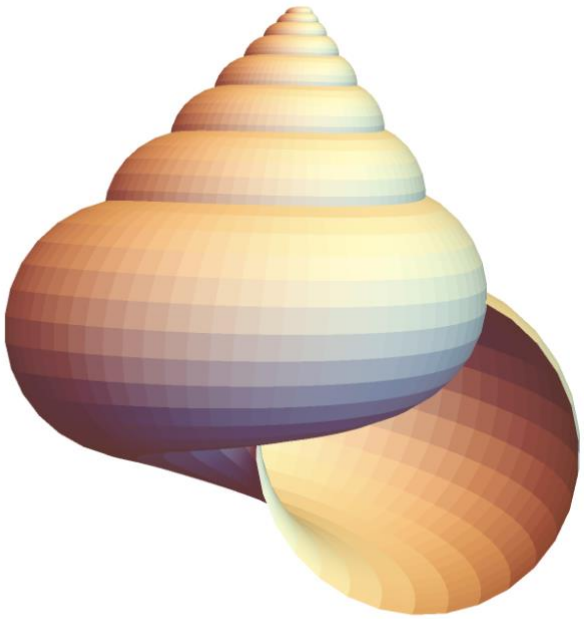
- *shape* is a matrix of 3D semilandmarks that circumscribe the aperture
- *W* is Raup's whorl expansion rate parameter
- *T* is Raup's vertical translation rate parameter
- *D* is Raup's parameter for distance of aperture from columella
- *TurnNum* is the desired number of whorls
- *True* is an optional argument that causes 3D point coordinates to be returned instead of a graphic

Example:

```
apert = {#[[1]], 0, #[[2]]} & /@ CirclePoints[30];

RaupCoil3D[apert, 1.5, 2, 0.3, 8]
```

Out[21]=



RaupCoil3DForPrint[*shape, W, T, D, TurnNum*]

Created 16 March 2016
Updated 12 July 2022 to add coordinate option.

This function generates a three-dimensional rendered shell using the functions developed by David Raup (1966, 1967).that is suitable for converting to solid mesh and exporting for 3D printing using the *SaveShell[]* function. It is nearly identical to *RaupCoil3D[]* except it returns a shell with double walls and a cap at its apex as 3D coordinates for use with the *SaveShell[]* function.

Arguments:

- *shape* is a matrix of 3D semilandmarks that circumscribe the aperture
- *W* is Raup's whorl expansion rate parameter
- *T* is Raup's vertical translation rate parameter
- *D* is Raup's parameter for distance of aperture from columella
- *TurnNum* is the desired number of whorls

Example:

```
apert = {#[[1]], 0, #[[2]]} & /@ CirclePoints[30];

myShell=RaupCoil3DForPrint[apert, 1.5, 2, 0.3, 8];
```

SaveShell[*myShell ,parameters ,title ,path*]

Created 16 March 2016.

This function saves a 3D shell suitable for printing as an OBJ mesh file .

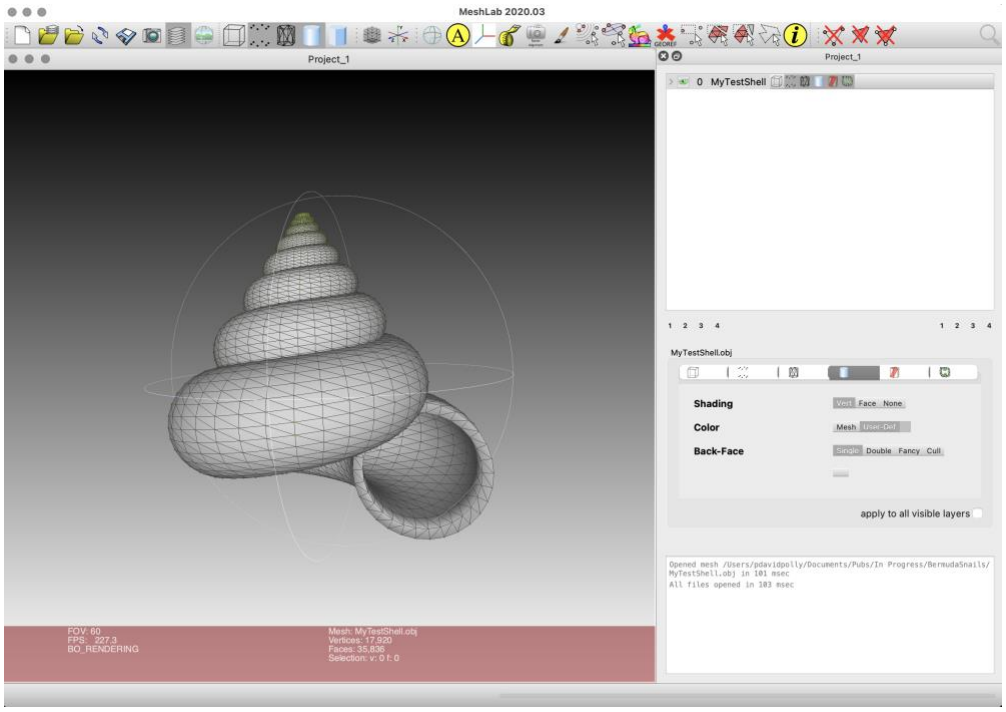
Arguments:

- *myShell* is a matrix of 3D coordinates that describe a double-walled shell like those produced by *RaupCoil3DForPrint[]*
- *parameters* is a list of the *W*, *T, D, and TurnNum* parameters that were used to generate the shell. These are written to the OBJ file as metadata to help with record keeping.
- *title* is the filename that will be used, which is also saved as metadata in the OBJ file
- *path* is the full path to the save location

Example:

```
myShell = RaupCoil3DForPrint[apert,1.5,2,0.3,8];

SaveShell[myShell,{1.5,2,0.3,8},"MyTestShell","/users/me/Documents/"]
```



Rendering of the saved OBJ file in MeshLab.

SnailsVersion[]

Prints the version number and citation for the current installation.

Example:

```
SnailsVersion[]

Snails for Mathematica 1.1
(c) P. David Polly, 15 October 2025
```

TiltAperture[*shape, angle*]

Created 16 March 2016.

This function tilts the aperture shape at a specified number of degrees. Normally Raupian shell models place the aperture in a vertical plane, but many snails have a tilted aperture relative to the axis of coiling. This function reorients the aperture shape for input into the coiling functions in this package.

Arguments:

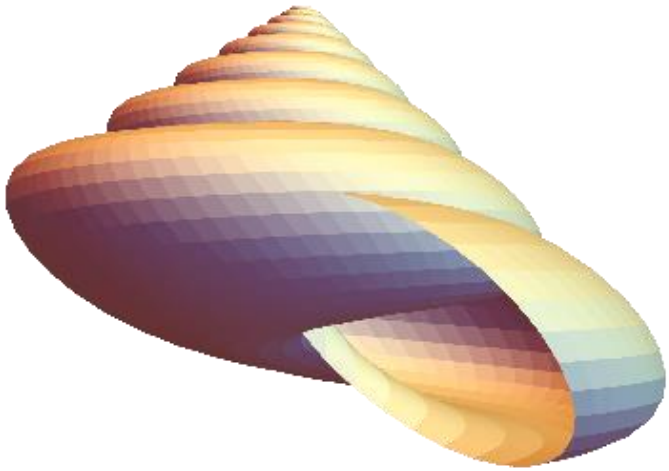
- *shape* is a matrix of 3D semilandmarks that circumscribe the aperture

- *angle* is an angle in degrees to tilt the shape relative to the axis of rotation.

Example:

```
myTiltedApert = TiltAperture[apert, -45];

RaupCoil3D[myTiltedApert,1.5,2,0.3,8]
```



Acknowledgements

Funding for development for this package has been provided by the Robert R. Shrock fund at Indiana University and the Yale Institute for Biospheric Studies.

Bibliography

Raup, D. M. 1966. Geometric analysis of shell coiling: general problems. *Journal of Paleontology* 40:1178-1190.

Raup, D. M. 1967. Geometric analysis of shell coiling: coiling in ammonoids. *Journal of Paleontology* 41:43-65.