

Shelly

COLLABORATORS

	<i>TITLE :</i> Shelly		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		June 24, 2022	

REVISION HISTORY

<i>NUMBER</i>	<i>DATE</i>	<i>DESCRIPTION</i>	<i>NAME</i>

Contents

1	Shelly	1
1.1	Contents	1
1.2	introduction	2
1.3	changes	2
1.4	contents	4
1.5	requirements	5
1.6	installation	5
1.7	quickstart	5
1.8	usage	6
1.9	general	6
1.10	datafiles	7
1.11	calcmmod	8
1.12	animation	10
1.13	problems	11
1.14	hints	11
1.15	algorithm	12
1.16	credits	13
1.17	distribution	13
1.18	disclaimer	14
1.19	adress	14
1.20	keywords	14

Chapter 1

Shelly

1.1 Contents

21.11.1994 ↔

Welcome to:

SHELLY V1.6

Introduction

Changes

Contents

Requirements

Installation

Quick Start

Usage

Algorithm

Credits

Distribution

contacting the author

Disclaimer

1.2 introduction

INTRODUCTION:

Shelly generates 3D-Objects of various seashells, slug-houses etc.
for: POV-V2.0, Real3DV2, X3D and T3Dlib
(the last means Imagine, DXF, Rayshade... support!).

It uses an
 algorithm
 found in:
Computer&Graphics Vol. 17, No. 1, pp. 79-84, 1993
("DIGITAL SEASHELLS" by M.B. Cortie.)

It was written in (portable) C using GCC2.6.3
and TDS, on an AMIGA.

The Real3D output is a RPL-Macro executable via "Execute named"
in the "Macros"-submenu.
The T3D-output can be converted via TDDD2xxx to many different
formats (of course you'll have to get the converters first)
(available on Aminet, look in 'pub/aminet/gfx/3d/' ...)

1.3 changes

Changes:

from V1.5 to V1.6:

- added simple animation mode ->
 Usage
 - more example pictures (pic/) + their datafiles (shy/) ↔
 available,
 mostly rendered with Real3D on my A4000
 - the exe does not need ixemul.library anymore
 (but shelly.ix !)
 - a new (POV-)output-mode uses Bezierpatches (Keyword: 'BEZ')
 - R3D2-output of NEWNOD-mode creates a "triset" instead of a bunch of
 triangles
 - added pic/parameters.iff (illustrates the use of some parameters),
 (took me some hours with PPaint ;)
 - Daniel Aregger programmed a nice GUI for Shelly
 (for the Amiga)
 this will be uploaded separately, look for FSG (Funky Shelly GUI)
-

- Due to the confusion Compuserve caused regarding the use of the GIF-format all GIF's are now IFF-ILBM's (If your viewer cannot handle this, use netPBM) sorry for this ...

from V1.4 to V1.5:

- changed NODULE-mode (consult the Usage section!) (some keywords are gone!)
- added new calculation-mode (NEWNOD)^^^!
- added new output (X3D, a VERY fast model-viewer for X11)
- included a small Tcl/Tk-GUI (start with 'wish -file sh.tcl' it should be self explanatory, so no documentation ... yet?)
- fixed some problems with not initialized data
- allows to set cameraposition for pov-output now (overrides internal calculation of the position)

from V1.3 to V1.4:

- fixed a bug in the NODULE calculation-mode (Shelly could end up in an infinite loop for some parameters)
- fixed a bug that caused "Float-Exceptions" (when calculating shells without nodules) on Alpha-Architecture (and others?)
- added "RENDER"-feature (Shelly calls POV itself to render a preview)
- added "Scale"-feature (the shell may be scaled now)
- Shelly now supports 3 different nodule-types in one shell (see threenod.shy)
- Shelly now produces outputfiles with the right versionnumber ;)

from V1.2 to V1.3:

- fixed a bug in the nodule creation (nodules were not symmetrical)
 - added new calculation mode (I call it "dynamic_stepsize"), please consult the Usage section!
 - added new outputtype ("RAW")
 - shelly uses a temporary file for calculation of the camera-
-

position (rather than calculating all points twice) now!

from V1.0 to V1.2:

- added "autofocus" for POV-output (automatic placement of the camera in the right distance)
- the file 'shelly.pov' is never consulted now!
- the RPL-output now generates much (100 times) smaller objects
- added T3Dlib-support (new Keyword is 'T3D'!)
- the silly countdown is gone
- this time Shelly comes with only one guide ! (hope so :))

1.4 contents

CONTENTS:

Shelly16.lha should contain:

- shelly (the executable for C= Amiga,
uses MC68020 and up & MC68881 and up)
 - shelly.ix (another exe that uses ixemul.library
and is slightly faster)
 - src/shelly.c
shelly.h (source & header, ready to compile on various
machines)
 - pics/ examples.jpg (a picture of some examples)
Ammonite.jpg
Turritella.jpg
... some more pictures
 - shy/ Planorbis.shy, Nautilus.shy, Lyria.shy, Ammonite.shy
Oxysteles.shy, Natalina.shy... (some example data-files)
 - Shelly.guide (this document)
Shelly.doc (^^^^ as plain text)
 - sh.tcl (a small Tcl/Tk GUI for shelly)
-

1.5 requirements

REQUIREMENTS:

Shelly requires atleast:

(to use the executable provided)

- an Amiga (harddisk & fast processor recommended)
- MC68882 (or up) FPU
- POV-V2.0 or Real3DV2 to look at the results,
or the TDDDlib-converters if you want Shelly to
create objects for Imagine etc.
- the .ix exe needs ixemul.library !

(on UX-Boxes)

- an ANSI-C compliant compiler (GNU)

Shelly has been tested on the following machines:

- A4000/030
- i486/Linux
- SUN4/10
- IBM/RS6000
- SGI/Indigo2

1.6 installation

INSTALLATION:

The installation of Shelly is very easy ...
Just copy the Drawer "Shelly" to a place
where you would like to install it.

and give it a '(g)cc shelly.c -o shelly -lm -O2' (not needed on Amiga)

1.7 quickstart

----- QuickStart:

To get started quickly :

- install Shelly (described in Installation)
- open a shell (CLI), cd to the directory "Shelly"
- type 'shelly shy/Planorbis.shy t:xxx.pov'
(Planorbis is one of the examples, xxx.pov is the name of the POV-Scene Shelly will create)
- now go and render the file 'xxx.pov'
(e.g. 'pov -ixxx.pov -f +d' (assuming you have pov in your path))

perhaps you have to edit the file 'xxx.pov' (camera position etc.) and try it again to get the best result...

For detailed information look into the Usage section.

1.8 usage

```
----- Usage:

General
Datafiles
Calculation modi
Animation
Problems
Hints
```

1.9 general

```
----- GENERAL

just type

'Shelly infile outfile'

to run Shelly from a shell (CLI)
```

- infile is the [path]name of a datafile
 - outfile is the [path]name of the POV/RPL/T3D/RAW/X3D ←
outputfile
- outfile will be overwritten (if it exists)!
- Shelly will also open a file "[path]outfilename.tmp"!
and overwrite it (this is a temporary file, it will be
deleted after calculation)

1.10 datafiles

DATAFILES:

Shelly uses own datafiles in a simple format to get
the arguments into the program.

Fileformat:

The files are of a very simple (and easy to process :)) format:

- every line of the file is scanned for
keywords
.
 - if a line contains no keyword it is treated like a comment
 - if a line contains a keyword it is interpreted
(the number/string behind the keyword is copied into an internal
structure: "alpha:30" sets the internal alpha value to 30)
(or a flag is set:
"RPL" sets the flag "we_have_to_produce_an_RPL-file")
 - the only "Flag-keywords" the program knows are: 'POV','RPL','T3D',
'RAW','X3D','NORMAL','NODULE','NEWNOD','RENDER','BEZ'
all other keywords need to be combined with a number or string
(as the ':' states)
 - everything is casesensitive! ('RPL' != 'rPl')
 - the file is not checked for anything else
 - double use of the same keyword causes an overwriting of the
last set value
 - lines like "alpha:Blafasel" will cause NO errormessage
(there are no messages at all ;))
-

note: the keyword does not have to stand alone!
if you write a line like:

```
"/*RPL*/" or "BlafaseRPLl"
```

the RPL-flag will be set! But you could also write:

```
"render this in RPL please :)"
```

There is a special file ("Blank.shy") prepared for you that is blank (contains only some necessary keywords).

consider:

- some parameters have to be given in degrees, some not (look into the algorithm section)
 - if you want Shelly to create a RPL file as output add a line like "RPL" or "please do it in RPL" to the file ("T3D" will switch to T3D-output) (POV output is default)
 - be careful with the parameters, don't try to fool Shelly ("what does it do if I enter an infinite value :)?") it will end up in a mess or coredump or our beloved friend because the values are not checked!
- You should just change the given examples slightly until you know what you are doing...
- o must be positive! (omin \geq 0, omax $>$ omin, od $>$ 0)

1.11 calcmold

CALCULATION MODI:

There are several ways of calculating a shell (means dividing the surface of the shell into subfaces)

currently 3 methods are implemented in Shelly:

The NORMAL-MODE:

just lays a grid (specified by smin, smax, sd, omin, omax, od) over the surface and calculates the knotpoints. Then the knotpoints are connected to triangles, or used to build a B-Spline-mesh.

The NODULE-MODE:

This is a new (changed in 1.5!!!) calculation mode that can save you object-data, up to 40%! (but only in shells with nodules!)

It is called "dynamic_stepsize" and can be invoked by using the NODULE keyword. (default mode is NORMAL)

How it works:

- imagine a shell with nodules (look at "nodule.iff"), if there is enough space between the nodules it might be possible to increase the stepsize (in o-direction) between the nodules without loss of information.
- this means high resolution is only used when needed (in the nodules) the rest of shell is calculated in a lower resolution
- The Shell is scanned in (O-direction) for nodules (with a stepsize given via new keyword 'Scano:'). In a nodule Shelly will compute the height difference to the next line. If the height difference exceeds a threshold (keyword 'Hdo:'), the stepsize will be made smaller just that it exactly matches the height difference. The smallest possible step is limited by 'Scano:' too!)
- Default value for Scano: is 0.05
Hdo: is 0.1
this should be best for most cases, but feel free to experiment, as smaller the Hdo: as finer the nodules are calculated.

The NEWNOD-MODE:

This mode is very similar to the NODULE mode, except that it does the same thing (refinement dependant on height difference) for the S-direction too. (use 'NEWNOD' to switch to it) This causes an irregular structure of the resulting grid, so that output as B-Spline-mesh is impossible (the RPL-output will create a triset instead)!

The keywords for the NODULE mode ('Scano:', 'Hdo:') are used aswell as two new ones ('Scans:', 'Hds:').

- Default value for Scans: is 0.05
Hds: is 0.1
this should be best for most cases, but feel free to experiment, as smaller the Hds: as finer the nodules are calculated.
-

1.12 animation

ANIMATION

Idea for animation mode came from Steve Enns.

It is useful for creating a sequence of objects of an 'evolving' shell (you could do the same with objectmorphing in some RT's).

You have to specify two different datafiles and a number.

Shelly will interpolate between the two files in number steps and create number objects!

example

```
in 'file1':
```

```
  Animate:file2           ;specifies filename of second datafile
  Steps:10                ;specifies number of steps in animation
  L:0.1
```

```
in 'file2':
```

```
  L:2
```

```
'shelly file1 out'
```

will create 10 objects (named 'out0' - 'out9') where the noduleheight grows from 0.1 to about 1.9 .

Last object (out9) will not be generated with L==2! (this is to allow a chain of sequences without double created objects)

To generate an object with L==2 you must call Shelly again.

('shelly file2 out10')

shy/ anim.shy & animend.shy are an example.

Of course 'L' is NOT the only parameter that can be used for interpolation (try out ;).

And of course, you may change as much parameters as you want between two files!

Note:

Shelly will take outputformat from the first file.

If the second file contains an 'Animate:' too, you have to restart Shelly after calculation of the first sequence.

1.13 problems

PROBLEMS

- it is nothing to be seen in POV:
probably the camera/light positions are wrong
take a look at the data in your pov-file and correct this
- POV tells me something from "degenerated triangles"
nothing serious, just some triangles with 2 points the same
- Imagine does not accept the Shelly-output
convert the output via 'readwrite' from the TDDD-Package
to the binary TDDD-format
- Real3Ds annoying "Stack full" message comes up everytime
a macro is executed:
 - change the RPL-stacksize (menu: Settings/RPL)
(increase the "Parameter Stack")
 - open a new RPL-window
 - type: '(path+)macroname" LOAD'

- strange numbers (NaN's) occur in the output:

Well this problem is known to me but no solution (sorry).

Since the algorithm is somewhat complex I really don't want to have to find out which combination of which parameters cause this.

It is also a problem of the sideeffects and (numerical) stability of the "mathematic" functions I call.

note: I suppose zeros are the source of all this
-> try to avoid them

1.14 hints

Hints

for Real3D-users:

- if your shell appears to be open in o-direction.
just close the u-direction in R3D2 (modify/freeform/open,close)
- if you want nodules in RPL-objects:

You should choose proper values of od and sd to see the nodules at all

(if you have nodules that are 10° wide (in o-direction) and \leftrightarrow you choose an od of 40° you will see probably no nodules!) (this is also important for the POV-output)

You should double the nodule height (L) for B-Spline objects to get the same height of the nodules as a POV-output!

- if you want to create a shell without nodules you may double the sd and od values for B-Spline objects without loss of quality in many cases

1.15 algorithm

The Algorithm:

In this section you will find more detailed information on the algorithm used by Shelly and on the parameters it uses.

There is a new picture (pic/parameters.iff) that illustrates some parameters (better than text I hope ;)

- The basic idea of the algorithm is to simulate a shell shape by rotating & moving (©ing) an ellipse (or a part of an ellipse, or any other curve (a cardioid)) around an axis. This will end up in some sort of spiral-shape.
- The shape produced will depend on many things like:
 - starting size/place/orientation of the ellipse
 - exact form of the ellipse (nodules)
 - how fast is the ellipse growing while rotating etc.
- you can find the exact formulas in the original article or in the sourcecode (too lazy to write them here again, they are very complex)
- here is a list of all parameters that shelly needs to generate a shell:
 - angular parameters (given in degrees):
 - alpha :equiangular angle of spiral
 - beta :angle between z-axis and line from aperture local origin to xyz-origin
 - phi :tilt of ellipse major axis from horizontal plane
 - omega :amount of azimuthal rotation of aperture
 - my :amount of "leaning over" of aperture
 - smin :angle at which aperture generating curve begins
 - smax :angle at which aperture generating curve ends
 - sd :stepsize in s-direction
 - omin :angle at which spiral begins
 - omax :angle at which spiral ends

```

od      :stepsize in o-direction

P      :position of nodule, in terms of angle s
W1     :width of nodule in s-direction
W2     :width of nodule in o-direction

-linear dimensions
A      :distance from main origin of aperture at o=0
a      :major radius (long axis) of ellipse at o=0
b      :minor radius (short axis) of ellipse at o=0
L      :height of nodule at o=0

-other
N      :number of nodules per whorl

- the parameters smin,smax,sd,omin,omax,od determine
  how many triangles (controlpoints) are generated
  (how smooth is the shell and how many whorls are generated)
  -> be careful with these: memory usage and filesize depend
      directly on this parameters

- the parameters alpha,beta,phi,omega,my determine the orientation
  of the ellipse before (and while) rotating

- the parameters A,a,b determine starting place and size of the
  ellipse

- the parameters P,N,L,W1,W2 determine number,size and place
  of nodules

```

1.16 credits

Credits:

Thanks to:

- the 3 people e-mailing me to keep on work on Shelly
- M.B. Cortie for his article "Digital Seashells"
- the people who ported GCC & CSH to the Amiga

1.17 distribution

DISTRIBUTION:

Shelly may be distributed FREELY via any media as long as:

- 1) The contents of the Archive remain unchanged.
- 2) No money (except a small copying fee) changes hand.

(Although Shelly is Freeware I won't reject gifts like money, chocolate, your latest piece of (gfx related) code etc.. My adress can be found under "contacting the author".)

1.18 disclaimer

DISCLAIMER:

This program comes with no warranty, either expressed or implied. The author is in no way responsible for any damage or loss that may occur due to direct or indirect usage of this software. Use this software entirely at your own risk.

1.19 adress

send
chocolate, money, your programs, bug reports etc.
to:

Randolf Schultz
Unter den Linden 51
19079 Mirow
GERMANY

FON: 0385/272066 (ONLY at the weekend!!!)

INTERNET: rschultz@informatik.uni-rostock.de

1.20 keywords

The following keywords are supported:

```
'alpha:'
'beta:'
'phi:'
'omega:'
'my:'
'smin:'
'smax'
'sd:'
'omin:'
'omax:'
'od:'
'P:'
'L:'
'A:'
'a:'
'b:'
'W1:'
'W2:'
'N:'
'RPL' (switches to RPL-output)
'POV' (guess)
'T3D' (hmm)
'RAW' (simply the coordinates of the created triangles)
      ('rawtopov' makes smooth triangles for pov from this !)
'X3D' (output for fast viewing in x3d on X11)
'BEZ' POV-output in NORMAL-mode that uses Bezierpatches

'NORMAL' (default calculation mode)

'NODULE' (switches to new
          calculation mode
          (dynamic stepsize)
          only useful when rendering shells with nodules!)

'NEWNOD' (switches to new
          calculation mode
          (dynamic stepsize)
          only useful when rendering shells with nodules!)

'Scale:' defines a scale factor for the shell (default is 1.0)

'RENDER' switches preview on (Shelly will automatically call POV
          after calculation)
          ONLY available when output-type == POV! and
          'pov' must be in the search-path!

'POVARGS:' defines arguments of the pov-call
            (default is "-f +d +w200 +h160")
            use of this keyword overwrites all default-arguments
            passed to pov!
            (be sure to specify a complete argument-string for pov)
            "-ixxx" is added automatically (don't use this!)
```

```
'P2:'      P of second nodule
'W12:'     W1 of second nodule
'W22:'     W2 of second nodule
'L2:'      L of second nodule
'N2:'      N of second nodule
'Off2:'    offset (in W2 (O) direction) between
           nodule1 and nodule2 (in degrees)

'P3:'      P of third nodule
'W13:'     W1 of third nodule
'W23:'     W2 of third nodule
'L3:'      L of third nodule
'N3:'      N of third nodule
'Off3:'    offset (in W2 (O) direction) between
           nodule1 and nodule3 (in degrees)

'Scano:'   stepsize for scanning the shell for nodules (in O-dir)
           & minimal possible stepsize! (default is 0.05)
'Hdo:'     defines maximal height difference between two lines
           (default is 0.1)

'Scans:'   stepsize for scanning the shell for nodules (in S-dir)
           & minimal possible stepsize! (default is 0.05)
'Hds:'     defines maximal height difference between two knotpoints in
           S-dir
           (default is 0.1)

'camx:'
'camy:'    x,y,z position of the camera for POV-output
'camz:'
```

NEW in V1.6:

```
'Steps:'  INT      Number of objects to create
'Animate:' FILE    switches to animation mode, uses FILE as name of
           datafile to interpolate to
'BEZ' s.a.
```

(note that the ':' belongs to the keyword! you can use
for instance the word 'alpha' with no risk in comment lines)

(The meaning of a special parameter (keyword) can be found
in the section about the
algorithm
.)