ShellyLibV2.3

Randolf Schultz (randolf.schultz@gmail.com) 30. May 2008

This is the documentation of ShellyLibV2.3, the ShellShapeGenerator.

Contents

1 About this Document

This document has been written using the SGML-Tools formatting system to generate files in a variety of text formats from one source file. There are plain ASCII, HTML and PostScript versions of this document prepared for you.

In addition, you can use the provided SGML-source to generate other formats.

Note, that the plain ASCII version probably misses a lot of formatting information.

2 Introduction

ShellyLib is a small set of C-programs for generating seashell or snail shapes. ShellyLib supports many common file formats, see section 8 (Output Formats), and has been designed to be as portable as possible.

In contrast to Shelly (the ancestor), ShellyLib is not Freeware but Shareware. Please read the file `"License"` in the top level of the distribution for further information.

ShellyLib contains a small library (`"libshelly"`) and a GUI called `"Shell Laboratory"`, in addition it is possible to compile a command line driven shell generator (`"shelly"`).

The `"Shell Laboratory"` is written in Tcl/Tk and C. It uses OpenGL for immediate rendering of the generated shells.

The algorithm used to generate the shells is an extended version of the algorithm by M.B. Cortie, found in Computer and Graphics, Vol. 17, No. 1, pp. 79-84, 1993, "Digital Seashells". See section 9 (Addresses, Pointers, Literature) for more references.

3 Changes

Changes from V2.2 to V2.3 are:

- General Changes:
  - New output format Wavefront OBJ.

  - The output format X3D now outputs Web3D X3D XML data.

- Changes to the Shell Laboratory:
  - Improved parameter names in the GUI.
  - Added wheel mouse bindings to the GUI.
  - Improved usability of generation curve editor (continuous insertion and deletion of points).
  - Added a texture macro recorder.
  - Texture macros/parameters are now also saved to ShellyLib data files.
  - There are now two adjustable background colors (for Wireframe and Shaded display modes).
  - Textures may now be imported from/exported to the TIFF image file format.

Changes from V2.1 to V2.2 are:

- General Changes:
  - A new parameter named `"alpha2"` has been introduced. This works much like alpha. It allows to adjust the growth rate of the generating curve without affecting the helico spiral. Oops? Read section 5 (Parameters of the Shell Generator) for more information.
  - The 3DMF (3D MetaFile) output format has been added; NURBS and triangles are supported.
  - The RIB output now contains own texture coordinates, that improve the texture mapping quality. This change is only visible for custom generating curves (GCURVE-mode) where the points are not uniformly distributed.
  - A new simple shader (SLtxtsrf) for RenderMan has been added, that maps the generated textures on the shells just like the `"Shell Laboratory"` does. Besides that, it shows how to use the custom texture coordinates of the RIB output.

- Changes to the Shell Laboratory:
  - Several changes to the hotkeys in the 3D-View and other places have been made.
  - A new RIB output mode (export scene) has been added. This mode saves all movements and rotations etc. you made in the 3D-View as well as the shell.
  - Texture-mapping quality in the 3D-View has been improved for the NURBS rendering mode. This change is only visible for custom generating curves (GCURVE-mode) where the points are not uniformly distributed.
  - Loading of values (via .shy-files) with higher resolution as the corresponding slider allows has been corrected. The sliders resolution is now recalculated according to the loaded value.
  - Added Antialiasing via viewing-volume-jittering to the rendering code. This improves overall rendering quality and works in all rendering modes, although it is not meant to be used with the wireframe mode. Two levels of antialiasing are available.
  - Added a status-bar, that keeps you informed about whats going on at the moment.
  - Added simple (single step) undo to the texture generator.
  - Added simple image operations (mix, add, sub) to the texture generator. The new texture may be combined with the older one using this operations.
  - Exporting the image of the 3D-View did not work when it was resized, fixed.

  - The ambient, diffuse, specular and background color of the shell/view may be changed separately now. Check the `"Color"` submenu of the `"Preferences"` menu.

Changes from V2.0 to V2.1 are:

- General Changes:
  - Due to the low registration moral (I received one (!) registration request, thank you!) the license changed again. Please read the file `"License"` in the top level of the distribution. Important changes are that I do not release the source for free anymore and I crippled the free available executables of the Shell Laboratory. Registered users can still get the full source, however.
  - `"libshelly"` now simulates that the walls of the generated shells have a certain thickness, new parameters are `"wall"` and `"wlen"` (see section 5 (Parameters of the Shell Generator) for more information).
  - Nodules are now possible in the `"GCurve"` mode.
  - The DXF output format has been added; Splines, polyface-meshes and 3dfaces are supported.
  - The RIB output now contains NURBS instead of B-Splines.
  - The RenderMan shaders have been rewritten. They do respect the new wall feature now.
- Changes to the Shell Laboratory:
  - The Shell Laboratory has been ported to WinNT/95.
  - The Shell Laboratory is now available as stand-alone executable that does not need a bunch of .tcl-files or an installed Tcl/Tk at all. Precompiled executables are available for Linux, WinNT/95 and Irix.
  - Several changes to the GUI (menu layout, hotkeys etc.) have been made.
  - The smooth-shaded rendering mode has been replaced by NURBS rendering using GLU.
  - Another rendering mode (NURBS+Wire) has been introduced. This mode displays a NURBS surface and the associated control polygon.
  - A generation curve editor has been added to the Shell Laboratory, see section 4.4 (The Generation Curve Editor) for more information.
  - A texture generator has been added, that utilizes the algorithms published in "The Algorithmic Beauty of Sea Shells" by H. Meinhardt (see section 9 (Addresses, Pointers, Literature) for a complete reference, and section 4.5 (The Texture Editor) for the documentation).
  - It is now possible to export the rendered image.

A lot has changed from version 1.6 of Shelly (the ancestor of ShellyLib) to version 2.0:

- Legalities have changed, read the file `"License"`!
- Almost everything has been completely rewritten, therefore, I suggest to reread the whole documentation, even if you feel familiar with Shelly!
- Bezier output for POV-Ray creates really smooth surfaces now.
- Several new output formats, e.g. Truespace, RIB, SCED, have been added.

- The handling of scale factors has changed. There are file format dependant scale factors now, to ensure a unique size of the generated shells over the variety of output formats.
- ShellyLib has a really cool GUI now, unfortunately, it is not as nearly as portable as the library itself.
- The former "NewNod" mode is now called "Nodule" and has been improved in terms of speed and usability. The old "Nodule" mode is gone.
- The code has been split into several parts. The shell generation is done by a small library called `"libshelly"`. Furthermore, there are two example frontends that use the library `"shelly"` and the `"Shell Laboratory"`.
- The original algorithm of M.B. Cortie is limited to ellipsoid generation curves. ShellyLibV2.0 allows the user to define any generation curve in a special new calculation mode called "GenCurve". This is useful for Cones or the "Miraculous Thatcheria", which were impossible to generate with the old algorithm.
- Some special keywords regarding POV-Ray are gone.
- Various other goodies have been included in the distribution, for instance displacement and surface shaders for RenderMan and a reference sheet that helps to understand the algorithm.

4 Using the Shell Laboratory

The `"Shell Laboratory"` is a GUI for `"libshelly"`. It is mainly a Tcl/Tk script and some C-code to interface to OpenGL (via Togl) and `"libshelly"`.

The main goal of this GUI is to ease the process of parameterization through immediate display of the results.

4.1 The Main Window

Each parameter of the shell generator is represented as a line, containing

- an entry (showing the value of that parameter)
- a configurable slider (it's minimum and maximum values are displayed in two labels to the right and left)
- a button to start the calibration of the slider.

Since all those parameter-lines fit hardly on any screen, I put them into a scroll-able canvas. You might want to maximize the window size to your full screen height. You can cycle through the entries and sliders with <`TAB`> and <`Shift+TAB`>.

Each change of a sliders value results in the following internal actions. At first, a new shell is generated via `"libshelly"` in the mode selected in the `"Preferences/SL-Mode"` sub menu. Then, the shell is displayed as wireframe, shaded polygonal or smooth NURBS model in the second window called `"3D-View"`. If you enter a value for a parameter directly in the associated entry, the update process starts when the focus leaves the entry (the <`TAB`>-key is pressed or another entry is selected with the mouse).

The initial configuration of the sliders has been choosen to protect you from potentially dangerous parameter values, that may crash the lab, and to ensure greatest possible freedom in setting any parameter. From time to time it may be necessary to reconfigure a slider, so that other values than allowed by its initial configuration may be set. Each slider may be configured using the associated `"Cal."` button. This will open a modal dialog, where you can change minimum and maximum value and, important, the step-size or resolution of the slider. Note, that you cannot set any other value than `x = min + (n * step-size), (x <= max)` with the slider.

4.2 The Main Menu

- `File`
  - `New`: reset the lab
  - `Load`: loads a set of parameters
  - `Save`: saves a set of parameters
  - `Import`
    - `Texture (PPM)`: import an image file in the PPM format to map on the shell
    - `Texture (TIFF)`: import an image file in the TIFF format to map on the shell
  - `Export`
    - `Shell`: export the calculated shell as object in various formats (see section 8 (Output Formats) for more information)
      - POV
      - DXF
      - . . .
    - `Scene`: export the calculated shell and all transformations of the 3D-View in RIB format
    - `Texture (PPM)`: export the current texture as image file in the PPM format
    - `Texture (TIFF)`: export the current texture as image file in the TIFF format
    - `3D-View Image (PPM)`: export the rendered shell as image file in the PPM format; Note, that the front buffer will be used for this, make sure that no other window obscures/destroys the content of this buffer. The resulting image files are usually very large, because the ASCII version of PPM is used, convert them!
    - `3D-View Image (TIFF)`: export the rendered shell as image file in the TIFF format; Note, that the front buffer will be used for this, make sure that no other window obscures/destroys the content of this buffer.
  - `Exit`: quits the application
- `Preferences`: (Configuring miscellaneous things)
  - `Mode`:
    - `Wireframe`: switches to wireframe display (default)
    - `FlatShaded`: switches to a flat (constant) shaded representation
    - `NURBS`: switches to a NURBS rendered representation
    - `NURBS+Wire`: NURBS + control polygon
  - `Projection`:

    * `Perspective`: Perspective projection (default)
    * `Orthographic`: Orthographic projection
  - `Color`: Change color and material properties of the shell
    * `Set Ambient Color`
    * `Set Diffuse Color`
    * `Set Specular Color`: color of the highlights
    * `Set Shininess`: specularity (0 - 128)
    * `Set Background Color`
  - `Antialiasing`:
    * `None`: disables antialiasing (default)
    * `4 Passes`: enables antialiasing using 4 rendering passes
    * `8 Passes`: enables antialiasing using 8 rendering passes
  - `DrawCoordsys`: switches display of the small coordinate system on or off (default: on)
  - `AutoUpdate`: If this is enabled, every change of a parameter results in an immediate update of the display. Toggling the checkbox from `"off"` to `"on"` will not result in an updated display. (default: on)
  - `Set NURBS Quality`: allows to change `"GLU_SAMPLING_TOLERANCE"`
  - `SL-Mode`:
    * `Normal`: (default)
    * `Nodule`:
    * `GenCurve`: see section 6 (Calculation Modi) for more information on these modes

- `Editors`:
  - `GenCurve-Editor`: opens the generation curve editor
  - `Texture-Editor`: opens the texture editor
- `Help`: (A small online help)
  - `Shell Laboratory`: Just a pointer to the original docs.
  - `About Shell Laboratory`

Shortcuts (accelerators) exist for some of these menu entries, see the menu entries for more information.

4.3 The 3D-View

This window uses OpenGL to draw the shell. The shell may be displayed as wireframe model or rendered using constant shading or as a NURBS-surface. You may configure this with the `"Mode"` sub-menu in the `"Preferences"` menu.

You may rotate, zoom and move the shell (moving along z-axis only) with your mouse and keyboard. See the listing of key and mouse bindings to learn how to.

A coordinate system is drawn to help you to navigate. It might be helpful to estimate the size of the object, as each axis is exactly 1 (whatever) long. It can be switched off with the `"DrawCoordsys"` check-button in the `"Preferences"` menu.

The view is updated every time you change the value of a parameter. You may change this behavior with the "AutoUpdate" check-button. Note, that changing "AutoUpdate" from off to on state does not result in an updated display.

If you get annoyed by the perspective distortions that may occur when viewing large objects, use the new orthographic projection mode (submenu "Projection" in the "Preferences" menu).

If you are lost in space use <r> to reset to the default settings.

4.3.1 NURBS

Some notes about the NURBS rendering mode. Be warned, that NURBS rendering may be a very lengthy process, especially if texturing is enabled. Under certain conditions it may last too long, even for the most patient. A simple rule to avoid long rendering times:

Do not render shells with NURBS that do not fit completely into the view.

Especially if you zoomed the view a lot, or if you have the impression that the viewer is inside or very near the shell (after a rotation or parameter change) do not activate NURBS!

Setting too small values for the "NURBS Quality" (which is actually "GLU_SAMPLING_TOLERANCE") is dangerous too, I recommend not to set values smaller than 10. However, I have to admit that you need to set it to 5 to correctly render some example .shy-files (namely, the rippled Cockle and Rapa).

Even though NURBS represent very smooth surfaces, occasional cracks might be visible on the surfaces, dependig on the OpenGL implementation. The NURBS rendered with Mesa usually do not suffer from this problem. On an SGI decreasing "NURBS Quality" or sometimes simply rotating the shell a bit helps.

The NURBS rendering speed of normal sized shells might not be enough for interactive displays on nowadays commodity workstations (Pentium class), thats why the representation is automatically switched from NURBS to flat shaded, while rotating and changing parameters with the sliders.

4.3.2 Antialiasing

Antialiasing improves the rendering quality using a technique called viewing-volume-jittering: the scene is rendered multiple times with slightly different camera settings. All passes are accumulated to the resulting image. Note, that the rendering times do not increase linearly with the number of rendering passes because of the buffer operations (the accumulation process). This means, if you get 10 frames per second in wireframe rendering mode, you will not get a fourth (2 fps) with 4 pass antialiasing, this depends on the speed of the buffer operations. In practice, expect something near 0.75 frames per second in wireframe mode! You see that antialiasing is very slow and therefore it is switched off temporarily while rotating and changing parameters with the sliders.

Two levels of antialiasing are available. The 4 passes mode increases the quality considerably, while 8 passes give not much more quality (compared to the increased effort). Your mileage may vary, however.

You may toggle antialiasing directly with the <a> hotkey in the 3D-View.

4.3.3 Texturing

Texturing can be enabled or disabled with the <t> hotkey. You may use imported textures or textures generated in the "Shell Laboratory" by the built-in texture generator.

Textures should be sized properly in order to be used with OpenGL. Use powers of two, otherwise the texture image will be scaled, which is something you might not want.

The "Shell Laboratory" uses mipmaps and bilinear filtering in order to achieve the highest possible texture mapped rendering quality (depending on the OpenGL implementation). Some strange blurry artifacts occurred lately using very large (2048x512) textures, I am still investigating this...

Note, that the white highlight on the shell is not rendered correctly if texturing is enabled. This is a limitation of OpenGL, which can be circumvented by a multi pass rendering approach (like most limitations of OpenGL), but I guess antialiasing is multi pass enough and there are programs called "renderer" for a reason...

4.3.4 Key and Mouse bindings

Here are all the key and mouse bindings:

<Mouse-Button-1> (the left one) click and drag: rotate around x- and y-axis

<Cursor-Left>, <Cursor-Right>: rotate around z-axis

<Cursor-Up>, <Cursor-Down>: move along z-axis

<Shift+Cursor-Up/Down/Left/Right>: move view

<x/X/y/Y/z/Z>: view along x (y z) axis

<+>: zoom in

<->: zoom out

<r>: reset all transformations

<a>: toggle antialiasing

<t>: toggle texturing

Other useful keybindings are <w> (switch to wireframe), <f> (switch to flat shading), <n> (switch to NURBS) and <c> (switch to NURBS + control polygon).

Note, that these key and mouse bindings are available in the 3D-View window only!

4.4 The Generation Curve Editor

The generation curve editor (GCE) offers a facility to easily create or modify a generation curve, used in the GenCurve-mode of "libshelly".

Some basics:

- The GCE always works on a local copy of the generation curve which is used by the main part of the Shell Laboratory. The original generation curve, which is used for generating shells, is updated only when you want it. (Check the "Update" menu.)
- Changes to the generation curve will only have an effect on the shell displayed in the 3D-View, if "libshelly" is in the "GenCurve" mode. The GCE will ask you to enable this mode on startup, in case you did not activate it by yourself.
- You may zoom and move the view of the generation curve with your mouse or keyboard, see keybindings.

- You may move, delete and insert points of the generation curve.
- Atleast two points are needed in a generation curve!

4.4.1 The GCE Menu

The menu items should be self explanatory.

- `Window`
  - `Close`
- `Update`
  - `No automatic Update`
  - `Update after move`
  - `Update while moving`
  - `Update now!`
- `Curve`
  - `Move Point`
  - `Insert Point`
  - `Delete Point`
  - `Fetch from Normal-Mode`
- `Help`
  - `Help`

4.4.2 Key and Mouse Bindings

Some accelerators exist for the menu:

<`Ctrl+w`>: close window

<`Ctrl+m`>: move points

<`Ctrl+i`>: insert points

<`Ctrl+d`>: delete points

<`Ctrl+f`>: fetch curve from Normal-Mode

<`Ctrl+u`>: update generation curve of the main window

Key bindings for zooming and moving are:

<`Mouse-Button-3`>: (the right one) click and drag: move the view

<`Up,Down,Left,Right`>: move the view

<`r`>: reset all moves

<`+,Add`>: zoom into the view x2 (works upto a factor of 8)

<`-,Sub`>: zoom out of the view x0.5 (works upto a factor of 0.125)

4.4.3 Inserting and Deleting Points

To insert a new point simply select the "Insert Point" menu entry, then click on a point of the generation curve. Do not release the mouse button, but simply drag the new point to it's position. Deleting points is even more simple, use the "Delete Point" menu entry and select the point to delete with your mouse. Note, that you cannot delete any points if there are only two left.

4.4.4 Fetching the Curve from Normal-Mode

If you just want to change some features of an ellipsoid generation curve you may fetch one easily from the shell generator for editing purposes. The curve fetched using the "Fetch from Normal-Mode" curve menu entry is the first curve that would be generated in the Normal-Mode. This means that certain transformations are applied to the original centered ellipsoid generation curve before fetching!

You may preview the curve you will get, by setting the following ShellyLib parameters "omin" to 0 and "omax" to 1 and by switching the shell 3D display to wireframe.

If you want to start your work on the generating curve with the same shell as would be generated in the Normal-Mode, set temporarily the following parameters:

"beta" to 90.0;

"phi","my","omega", "A" to 0.0;

"a", "b", "scale" to 1.0.

Now fetch the generating curve, and restore the original values. This does not give you exactly the same shell, but the changes should be minimal.

4.5 The Texture Editor

The texture editor (TE) offers a facility to easily create textures to be mapped onto the shell shapes generated by "libshelly". It utilizes the algorithms published in "The Algorithmic Beauty of Sea Shells" by H. Meinhardt (see section 9 (Addresses, Pointers, Literature) for a complete reference). This section is not intended to explain these algorithms and their parameters in depth. If you are interested read the book.

Some basics:

- The TE always works on a local copy of the texture which is used by the main part of the Shell Laboratory. The original texture, which is used for rendering shells and texture export, is updated only when you want it. Check the "Update" menu.
- Changes to the texture will only have an effect on the shell displayed in the 3D-View, if texturing is enabled. The TE will automatically activate texturing when you update the texture.
-

4.5.1 The TE Menu

The menu items should be self explanatory.

- Window

  - Close
- Macro
  - Clear
  - Remove Step
  - Run
  - Record
- Update
  - No automatic Update
  - Update after generate
  - Update now!
- Help
  - Help

The following hot-keys are defined:

<Ctrl+w>: close window

<Ctrl+u>: update texture

<Ctrl+g>,<g>: generate texture

<Ctrl+s>,<s>: stop generator

<Ctrl+z>: Undo

4.5.2 Generating a texture

For your first experiments just select a texture type using the "Select type:" listbox and use the "Generate" button.

Follow these steps to create a texture:

1. At first, set the desired size of the texture. If you plan to map the generated texture onto a shape in the Shell Laboratory, you should stick to sizes that are powers of two (64, 128, 256 etc.), as OpenGL demands this. GLU does scale the texture to the next power of two, but this might not deliver best results. This scaling does not affect the size of any exported textures. Note, that some OpenGL implementations impose even more constraints (a maximum texture size of 256x256 for instance). Also note, that some texture type specific parameters may need to be adjusted, if you select a different height than the default height of 128.
2. Now select the type of the texture, with the "Select type:" listbox. Selecting a type will fill the next two sections of adjustable parameters with default values.
3. You may alter the type specific settings now. Set colors and adjust parameters of the texture generator, more information about these settings can be found in the next section: 4.5.3 (Parameters of the texture generator).

4. Choose an image operation now. The default (Overwrite) simply destroys the old texture.
5. Now you can generate the texture, using the "Generate" button. If you find that the created texture does not come out right, you might stop the generator with the "Stop!" button. Note, that while generating a texture, the GUI might respond a little slower than usual.
6. The generated texture can be mapped onto a shell now. Simply press "Ctrl+u", then change to a Flat-shaded or NURBS display. Texture mapping may reduce performance drastically on software-only OpenGL implementations; switch to the wireframe display before rotating or changing parameters!

4.5.3 Parameters of the Texture Generator

How does the texture generator work?

It simulates the distribution of substances (e.g. pigments) in the cells of a shell over it's lifetime. Several different effects may influence the distribution of substances like diffusion. All these different effects are put together in sets of equations. Each set of equations allows for the simulation of a specific type of shell pattern. There is no general set of equations available that could simulate all possible patterns of shells.

It is clearly beyond the scope of this documentation to explain all the equations and their parameters, as they are really complex. I can only point you to Meinhardt's book.

The texture generator implements many (not all!) equations from Meinhardt's book "The Algorithmic Beauty of Sea Shells". The parameter names were taken from the example implementation given in the book.

Some features of Meinhardt's software are not implemented in the texture generator. It is not able to plot concentrations of different substances, but just one (substance A). The type of the plot is always a continuous mapping of the concentration of substance A to color values. This mapping can be adjusted with the "Set Color:" parameters. They define an interval of concentration values (the two entries) and the accompanying interval in color (the two colored buttons). Two more things are important about this mapping: if the concentration of substance A is smaller than the value in the first entry, the resulting color is white; if the concentration is bigger than the value in the second entry the resulting color will be that of the second button.

Now on to the other parameters. You may select different parameter sets with the pop-up-menu that comes up if you press the "Set Parameter:" menu button.

Up to seven substances (named A to G) may be used for the simulations. That's why there are seven numbers in each parameter list (except for the "Misc."-section). The first number in the list (if "Initial Concentration (A)" is selected) is the initial concentration of substance A, consequently. The second number is the initial concentration of substance B and so on. Do not confuse the names of substances (A to G) with the names of parameters (A, B, C, D, G, R, S).

An important section of parameters is the "Misc."-section. The parameter names from this list (except for "RS") have been taken from the example implementation by Meinhardt as well.

Some simulations depend on random numbers. You can set the amount of randomness with the "KR" parameter. The "RS" parameter controls the initialization of the random number generator. If "RS" has a value other than zero, this value will be used to initialize the random number generator. This is done before each simulation run, this way the results of random simulations become predictable. If you set "RS" to zero, the random number generator will not be initialized at all. The resulting pattern is not predictable anymore, and you may generate different patterns just by pressing the "Generate" button again and again.

For the other parameters, please consult the book.

Note, that not every set of equations uses all seven substances. Nevertheless, you must have seven numbers in each parameter section (except for the "Misc."-section of course).

Here is a short survival-guide to help you to get certain things done:

- Problem; Solution
- The pattern is to small/big in O-direction; Increase/decrease "KP" (might not work for all texture types, as this actually controls the number of iterations used to calculate a single column)
- The pattern is to small/big in S-direction; Increase/decrease the "Height"-value.
- The texture generator seems to stop before the end, leaving a lot of white space; If the "wall"-parameter has a value higher than "0.0" the texture generator will smoothly fade generated textures to white on the texture position that corresponds to the beginning of the aperture of the shell. This way the interior of the shell is not textured at all. If you change "wall" to "0.0" this will not happen.
- Texture comes out distorted or noisy; Sometimes simply generating a new texture will help (if you did not fiddle with any parameters it will), otherwise try to decrease "KR" (randomness) or the diffusion-constants, which should not be higher than "0.4" to avoid numeric instabilities.

4.5.4 Image Operations

Image operations may be used to combine generated images, this boasts the number of possible patterns considerably. You may choose between three effects, mix, add and sub; they should be self explanatory, if not: simply try them out. Press <Ctrl+z> (single step undo) if you are not satisfied with the results.

Use the associated entry to determine the strength of the effect.

The two source images should have the same dimensions, unless you want funny results.

4.5.5 Macro Recorder

Since ShellyLib2.3 there is a macro recorder integrated into the Texture Generator.

If recording is enabled, every completely generated texture creation pass is recorded with all its parameters (including texture size and image operation) as a macro step. If the "Stop!" button is used, no macro step will be recorded. All macro steps or just the last macro step may be erased using the menu entries "Clear" and "Remove Step" respectively. Using undo (<Ctrl+z>) will also remove the last macro step. A recorded macro may of course also be run using "Run", all recorded steps will be executed in the order of their recording.

Furthermore, a recorded macro will be saved to ShellyLib data files, written by the Shell Laboratory. If a loaded data file contains a texture macro, the Shell Laboratory will offer to run it immediately.

4.5.6 Sundial Tutorial

This section shows how the texture for the Sundial shell on the ShellyLib Home Page was created.

1. Load the "Sundial.shy" ShellyLib data file from the shy directory.

2. Open the texture editor.
3. Set the texture width to 1024.
4. Choose the "IrregularStripes" texture, generate it.
5. Now we want to add the perpendicular white stripes to the texture. Choose the "Stripes3" texture.
6. Set the the start value to 0.0, start color to black (0, 0, 0). Set the end value to 0.5 and the end color to a middle gray (128, 128, 128). This makes the stripes a bit broader(start - end values) and limits the generated color values to 128. For sharper white perpendicular stripes experiment with higher end values. Use "Ctrl+z" to undo the last change to the generated texture, if you are unsatisfied with the result!
7. Set "KP" from 20 to 30 ("Set parameter:/Misc"). This increases the number of stripes.
8. Switch to image operation "Brighten", change the strength of the effect to about 80 percent.
9. Generate again. The brown stripes should now be interrupted by white perpendicular ones.

5 Parameters of the Shell Generator

The basic idea of the algorithm is to simulate a shell shape by rotating a growing ellipse around the z-axis, and simultaneously displacing it from the z-axis. This results in a flat spiral. If we move the ellipse along the z-axis while rotating, a shell like shape will appear. The curve determined by all centers of all ellipses is a so called helico-spiral.

The parameters of this process are divided into 5 groups. Parameters that need to be given in degrees are marked with an asterisk ("*").

Angular Parameters:

These parameters change the shape of the helico-spiral ("alpha"* and "beta"*) and the tilt of the ellipse against the axes of the local coordinate system of the ellipse ("phi"*, "my"*, "omega"*). Note, that "alpha"* does determine the growth rate of the ellipse too.

"alpha2"* is a new parameter that has been introduced with V2.2. It works much like "alpha"* but it only changes the growth rate of the generating curve. At its default setting of 90.0, no change to the original growth rate takes place. At a higher value it will decrease the growth rate determined by "alpha"* and vice versa. Due to the dependence from "alpha"* you have to find a new value for "alpha2"* every time "alpha"* is changed! See the new "Snail.shy" for an example.

Linear Parameters:

They change the size (and shape) of the ellipse ("a", "b") and the distance of the first ellipses center from the z-axis ("A").

Dimensional Parameters:

They set how many ellipses are calculated ("omin"*, "omax"*, "od"*), and how many points on each ellipse are calculated ("smin"*, "smax"*, "sd"*). Finally, a global scale-factor, independent from all file format specific scaling, might be applied using "Scale".

Nodule Parameters:

Change all aspects of nodules, position on the ellipse ("P"*), size ("W1"*, "W2"*), length ("L"), number of nodules per whorl ("N") and starting point on the spiral ("Nstart"*).

There are three independent groups of nodules. Add a "2" to each nodule parameter to set parameters of the second nodule group. Length of second nodule is therefore "L2". Do the same for the third nodule, with a "3". There are two more parameters to control an offset between the three groups of nodules, "Off2"* is the offset between nodule 1 and nodule 2 and "Off3"* is the offset between nodule 1 and nodule 3.

Nodule-Mode Parameters:

"Scano", "Scans", "Hdo", "Hds" refer to section 6 (Calculation Modi) for information on these parameters.

Wall Parameters:

"wall" defines the thickness of the wall in terms of "a" and "b" and "wlen"* defines how many degrees into the interior of the shell the wall should be computed. A simple trick allows the calculation of the inner walls:

At the aperture of the shell "a" and "b" are multiplied with "wall" and then the shell generator computes the interior with negative "od".

The wall feature is a big win for realism, now even direct views of the aperture of the shell do not reveal the synthetic nature of the shell easily.

Unfortunately, simple scaling of the generating curve does not yield best results in every case, namely the GCURVE-mode. For best results, try to center the custom generation curves.

The most influential parameters wrt. the shell shape are "alpha", "beta", "A", "a" and "b".

Here are some hints on how certain values change the shape of the shell.

- Alpha 90 means no growing of the ellipse and no spiral shape at all. If beta is 90, too, this will produce a torus like shape. If alpha gets smaller the spiral will form and the ellipse will grow.
- Beta 90 means no moving of the ellipse along z-axis. This is fine for shells like "Nautilus", "Planorbis", "Ammonite" or several cockles. If you want to shape a "Turritella" or "Natalina" you have to decrease beta.
- The meaning of a certain value of A depends highly on a and b. Generally, A should be higher as a and b, otherwise the shell may look more like a sphere.

6 Calculation Modi

This section documents special calculation modi of ShellyLib.

6.1 The Nodule Mode

The parameters "od" and "sd" determine the resolution of the generated shell. But if the shell contains nodules that are thin and long you would have to use very small "od" and "sd" values in order to exactly

capture the shape of the nodules. This would lead to a real waste of memory, because between the nodules the high resolution is not necessary. To minimize memory complexity, a special nodule calculation mode has been introduced. This mode adapts both "od" and "sd" while calculating the shell, so that small distances between the calculated points are only used when needed, in the steep regions of the nodules. This process may be controlled with the parameters "Scano", "Scans", "Hdo" and "Hds". "Scano" determines the smallest possible distance of two points in O direction, the same does "Scans" for the S direction. "Hdo" defines the smallest possible difference in height between two points in O direction, the same does "Hds" for the S direction. The difference in height is being calculated using the height of the nodule. The nodule mode may be invoked by the "NODULE" keyword.

"Hdo" and "Hds" are multiplied by the nodule height of the first nodule group, before they are passed to "libshelly", when using the example frontends. Thus, the height difference value is depending on the nodule height. It is now easier to estimate the effect of a certain value, e.g. 0.1 means a tenth of the nodule height, no matter how high the nodule actually is. Note again, that this multiplication is not done by "libshelly".

Unfortunately, the result of this mode is of irregular structure. This means the only possible output type is triangles. No spline output, neither B-spline nor Bezier patches, may be generated using this mode.

6.2 The GenCurve Mode

The original algorithm of M.B. Cortie is limited to ellipsoid generation curves. ShellyLibV2.x allows the user to define any generation curve, using the "GenCurve" mode. This is useful for Cones or the "Miraculous Thatcheria", which were impossible to generate with the old algorithm.

The curve needs to be specified as a list of points with the "GCx:" and "GCy:" keywords. The parameters "a" and "b" still scale the curve as they do it in the other modes with the ellipse, and "A" moves the curve. However note, "smin", "smax" and "sd" have no influence on the shell shape in "GenCurve" mode, the generation curve completely takes over these parameters!

Nodules are possible within this mode, but the result depends highly on the number of points of the generation curve.

Note, that the resolution of the shell depends directly on the list of points. The spaces between the points are not interpolated. The curves provided in the example data files in the "shy" directory are meant to be used with spline output formats like "RPL" or "RIB", because the curves contain only few points. Note that not every output format is possible within this mode at present. You may use: "RPL", "RIB", "BEZ", "SCB".

Switch to this calculation mode with the "GCURVE" keyword.

7 File Format

ShellyLib uses files in a simple format to store sets of parameters. You should use a characteristic filename extension like ".shy" for these files. However, this is not mandatory.

Every line of the file is scanned for keywords. Lines that start with a hashmark ("#") are fully ignored. If a line contains no keyword it is treated like a comment.

Otherwise the number behind the keyword is copied into an internal variable or a flag is set. This implies, that there are two types of keywords, so called number-keywords and flag-keywords. Number-keywords need

to be combined with a number, flag-keywords have just to be there to set something. All names of the number-keywords correspond to parameters of the shell generator.

Number-keywords contain a ":" to remind you to write a number behind them. Flag-keywords are written in capital letters.

Keywords to change the so called "angular parameters" are "alpha:", "beta:", "phi:", "omega:", "my:".

Keywords to change the "linear parameters" are "A:", "a:", "b:".

Keywords for "dimension parameters" are "Scale:", "smin:", "smax", "sd:", "omin:", "omax:", "od:".

The keywords for nodule 1 are: "P:", "L:", "W1:", "W2:", "N:".

Nodule 2: "P2:", "L2:", "W12:", "W22:", "N2:", "Off2:".

Nodule 3: "P3:", "L3:", "W13:", "W23:", "N3:", "Off3:".

Keywords for wall thickness: "wall:" and "wlen:"

For an exact description of the parameters read section 5 (Parameters).

Keywords for switching output formats are "RPL", "T3D", "RAW", "X3D", "BEZ", "RIB", "SCD", "SCB", "TSP", "DXF", "DXC", "DXP". See section 8 (Output Formats) for more information regarding the formats.

Keywords for special calculation modi are "NORMAL", "NODULE", "Scano:", "Hdo:", "Scans:", "Hds:", "GCURVE", "GCx:", "GCy:". See section 6 (Calculation Modi) for more information.

Some remarks: Everything is case sensitive ("rpl" is not the same as "RPL")! Double use of the same keyword causes an overwriting of the last set value. Only one keyword per line is allowed. Maximum line length is 255 characters.

8 Output Formats

ShellyLib is able generate files in the following formats:

- Format, Program, Output-Keyword, file type (content), remarks.
- POV, **POV-Ray**, "BEZ" / "POV", Bezier Patches, Does not work in Nodule-mode! For Nodule-mode use "RAW" and **RAW2POV**.
- RPL, **Real3D**, "RPL", executable RPL-macro, creates a B-Spline-mesh in Normal-mode/GenCurve-mode or a Triset (triangles) in Nodule-mode.
- RIB, various, "RIB", RenderMan Interface Bytestream creates NURBS. Does not work in Nodule-mode.
- 3DMF, various, "3DM" / "3DN", 3D MetaFile (ASCII) NURBS ("3DN") or triangles ("3DM"), NURBS do not work in Nodule-mode.
- DXF, various, "DXF" / "DXC" / "DXP", B-Splines ("DXF"), PolyFace-Meshes ("DXP") or 3DFaces ("DXC"), "DXF" and "DXP" do not work in Nodule-mode.
- T3D, **T3DLib** / **Imagine**, "T3D", triangle mesh, Note, that Imagine cannot directly read these files, convert them from ASCII to binary using T3DLib first.
- RAW, **RAW2POV**, "RAW", triangles,

- COB, **Truespace**, "TSP", triangles
- SCD, **SCED**, "SCD" / "SCB", contains a triangle mesh ("SCD") or Bezier patches ("SCB"), "SCB" does not work in Nodule-mode.
- X3D, **X3D**, "X3D", triangles, For fast previewing (wireframe) using X11.

For more information regarding compatibility issues (will program xy read the file?) refer to the compatibility section of the ShellyLib Home Page: "http://www.shelly.de/".

8.1 Surface Types

ShellyLib allows you to save the generated shells in a variety of different representations, depending on the output format you have choosen (and sometimes additionally from the calculation mode). It is important to know some things about the different surfaces, and how they are generated:

- Bezier-patches result from "POV" or "SCB" output mode. POVRay and SCED do only allow patches that contain 16 controlpoints, four in each direction. Bezier surfaces interpolate only the four controlpoints on the corners of the patch. Hence, it is not possible to simply write Bezier-patches in a way that each point calculated by the shell generator becomes a controlpoint of the Bezier-patches. This would result in discontinuities and even holes at patch boundaries. Instead, new controlpoints have to be calculated so that the transition between two patches is smooth. For each four points of the original data from the shell generator a Bezier-patch with 16 controlpoints is being calculated. This may increase filesizes considerably! All generated points at the boundaries of the shell will be discarded. All other points generated by ShellyLib will be interpolated by the resulting surface. The algorithm used to calculate the controlpoints may deliver bad results in highly curved regions (thin long nodules, for example). You need to generate atleast four points in each direction to write a file in "POV" or "SCB" format! Even though a single Bezier-patch is built from four generated points, atleast 16 points are needed to write a single patch (four in each direction). This is because the calculation of the controlpoints of the Bezier-patch needs information about the surrounding patches.
- B-Splines result from "RPL" output mode if ShellyLib is in Normal-mode or GenCurve-mode. The resulting surface does not interpolate any generated points, and is much smoother than the Bezier-output. You can make Real3D interpolate the outer points of the mesh using "triple ends" in the "Modify/Freeform" menu (of Real3D, that is). It may be necessary to increase the length of nodules (parameter "L:", "L2:" and "L3:") to achieve proper nodule heights.
- NURBS result from "RIB" output mode if ShellyLib is in Normal-mode or GenCurve-mode. The resulting surface does only interpolate all the endpoints of the surface in S-direction. This is achieved by high knotvalues for these points. This has been done to ease the work with the generation curve editor (see section 4.4 (The Generation Curve Editor)). This way start and end point of the specified generation curve are touched by the surface. The endpoints in O-direction are not interpolated in order to achieve a smooth round surface. Atleast four points in each direction are needed. The shells rendered with GLU-NURBS and the shells rendered using the "RIB" output should look exactly the same.

9 Addresses, Pointers, Literature, Acknowledgements

The author may be contacted via e-mail: "randolf.schultz@gmail.com".

or snail mail:

```
Randolf Schultz
Unter den Linden 51
19079 Mirow
Germany
```

New versions and additional information may be obtained via the ShellyLib Home Page: "http://www.shelly.de/".

The following literature has been of much help

- Digital Seashells, M.B. Cortie, Computer and Graphics, Vol. 17, No. 1, pp. 79-84, 1993
- Shells, S. Peter Dance, Harper Collins Publishers 1992, ISBN 0 7322 0067 9
- The Algorithmic Beauty of Sea Shells, Hans Meinhardt, Springer-Verlag Berlin Heidelberg 1995, ISBN 3 540 57842 0