*<name>*
*<College>*
*<CRSID>*

Diploma in Computer Science Project Proposal

# Natural Object Modelling

*<date>*

**Project Originator:** This is a Model Project Proposal

**Resources Required:** See attached Project Resource Form

**Project Supervisor:** *<name>*

**Signature:**

**Director of Studies:** *<name>*

**Signature:**

**Overseers:** *<name>* and *<name>*

**Signatures:** *<no need to obtain Overseers' signatures yourself>*

*<This proposal refers to the language Java and to sea shells as an example of a natural object whose form can be closely described by algorithm. Java and sea shells are used only as examples. It is perfectly possible to use a different language and to model different natural objects. Likewise other specific features described may be taken simply as examples.>*

# Introduction and Description of the Work

This project is intended to focus on the process of generating data for computer graphics. Natural phenomena have posed many problems, and are still an area of great interest. Sea shells, with their semi-geometric forms, have been studied, and algorithms for automatically generating them proposed [1].

This project would involve implementing these algorithms for the generation of sea-shell models. Once the shells have been generated, they can be output in a form suitable for input into a rendering system. *<The renderer that is to be used must be identified here and* POVray (`www.povray.org`) *is a possibility.>*

# Resources Required

A ray tracer will be required. The public domain ray tracer *POVray* is available for download from `www.povray.org`. Output images can be viewed using a standard viewing tool.

# Starting Point

*<This is the place to declare any prior knowledge relevant to the project. For example any relevant courses taken prior to the start of the Diploma year.>*

# Substance and Structure of the Project

An input language will be designed to describe the different forms of sea shells and the parameters used to control their generation.

*<Proposed extensions should be described next, for example...>*

As an extension, an interactive user interface will be written that allows the selection and editing of parameters and families of shapes interactively. If time allows, some means by which the user may specify the patterning or texturing of the shells will be implemented. This might be based on some kind of texture generating function.

The mathematical descriptions in the paper will need to be implemented algorithmically, and the surfaces translated into a form suitable for graphical processing. Finally, this information will be output in a suitable form for the renderer.

The project has the following main sections:

1. A study of the algorithms involved in generating sea shells and an investigation of the workings of a suitable ray tracing package. A possible starting point is the 1996 Diploma Dissertation *Modelling Semi-Geometric Forms*, by Melvin Carvalho. *<List relevant references such as those found in this dissertation.>*

2. Developing and testing the code for the generation of sea-shell models, and for outputting these as files for input to the ray tracer. Potentially also code for an interactive user-interface or for user-defined texturing.

3. Evaluation using a variety of input parameters and preparation of example results to demonstrate that the implementation has been successful.

4. Writing the dissertation.

If time allows, there is scope for augmenting (2) with an interactive previewer that allows the user to examine the shape of the shells in a wire-frame view, so that the effect of parameter changes can be seen immediately.

# Variations

*<This section will not appear in your project proposal. It details some alternative ideas to generating sea shells. Such ideas would require rewriting the proposal to remove references to sea shells and insert references to plants or houses as appropriate.>*

As well as sea shells, there has been a lot of interest in the automatic generation of plants [2][3]. This has mainly been through the use of L-systems, a grammar-based approach that evolves plants through a series of random steps of re-write rules. The grammar symbols are finally translated into a three-dimensional description that can again be used as input into a rendering system. This project would involve several more aspects than the above, since an L-system engine, as well as a symbol to model translation would be needed.

A further variation might be the automatic generation of houses, following the paper given in [4]. Set the task of automatically generating models representing Victorian houses of different sizes, algorithms are described for the automatic positioning and sizing of rooms and walls. These are again grammar based, although other systems would be possible, based perhaps around systems such as Prolog.

# References


[1] *Modelling Sea Shells*, D. Fowler, P. Prusinkiewicz, J. Battjes, *SIGGRAPH '92 Conference Proceedings*, ACM Computer Graphics Vol. 26 No. 2, pp. 379–387.

[2] *The Algorithmic Beauty of Plants*, P. Prusinkiewicz, A. Lindenmayer, Springer Verlag 1990 (reprinted 1996).

[3] *Developmental Models of Herbaceous Plants for Computer Imagery Purposes*, P. Prusinkiewicz, A. Lindenmayer, J. Hanan, *SIGGRAPH '88 Conference Proceedings*, pp. 141–150.

[4] *Generative Geometric Design*, J. Heisserman, *IEEE Computer Graphics & Applications*, March 1994, pp. 37–45.


# Success Criteria

The following should be achieved:

- Implement and demonstrate at least one algorithm for generating seashells
- Store and display seashell models
- Output models in a format so that they can be input by a ray tracer and show example renderings produced
- Either: (*a*) implement and demonstrate a more advanced seashell generation algorithm or (*b*) implement a graphical user-interface for seashell design

# Timetable and Milestones

*<In the following scheme, weeks are numbered so that the week starting on the day on which Project Proposals are handed in is Week 1. The year's timetable means that the deadline for submitting dissertations is in Week 34.>*

*<In the Project Proposal that you hand in,* actual dates *should be used instead of week numbers and you should show how these dates relate to the periods in which lectures take place. Week 1 starts immediately after submission of the Project Proposal.>*

*<The timetable and milestones given below refer to just one particular interpretation of this document. Even if you select exactly this interpretation you will need to review the suggested timetable and adjust the dates to allow as precisely as you can for the amount of programming and other related experience that you have at the start of the year. Take*

*account of the dates you and your Supervisor will be working in Cambridge outside Lecture Term. Note that some candidates write the Introduction and Preparation chapters of their dissertations quite early in the year, while others will do all their writing in one burst near the end.>*

## Before Proposal submission

*<This section will not appear in your Project Proposal.>*

Submission of Phase 1 Report Form. Discussion with Overseers and Director of Studies. Allocation of and discussion with Project Supervisor, preliminary reading, choice of the variant on the project and language *<Java in this example>*, writing Project Proposal. Discussion with Supervisor to arrange a schedule of regular meetings for obtaining support during the course of the year.

Milestones: Phase 1 Report Form (on the Monday immediately following the main Briefing Lecture), then a Project Proposal complete with as realistic a timetable as possible, approval from Overseers and confirmed availability of any special resources needed. Signatures from Supervisor and Director of Studies.

## Weeks 1 to 5

*<Real work on the project starts here (as distinct from just work on the proposal). A significant problem for Diploma candidates is that this critical period largely coincides with the Christmas vacation. There is no guarantee that supervisors will be available outside Lecture Term, but Diploma students take much less of a Christmas break than undergraduates do, and so have some opportunity for uninterrupted reading and initial practical work at this stage. It is important to have completed some serious work on the project before the pressures of the Lent Term become all too apparent.>*

Study of sea-shell generation algorithms including some short test programs to test understanding of Java and the basics of the sea-shell algorithms. Experiment with ray tracer to get a feel for its capabilities. Write outline Introduction and Preparation chapters of dissertation.

Milestones: Some example Java code, which will probably not be used in the final project, and some example ray tracings from files written by hand.

## Weeks 6 and 7

Implementation of code for carrying out sea-shell generation and output of this in a simple form for the ray tracer. Preliminary ideas for more advanced features. *<To make a*

*reasonable project proposal you will need to implement at least one of: texturing, graphical user-interface, more complex sea-shell generation algorithms.>*

Milestones: Working code for sea-shell generator and output to ray tracer. Ideas for advanced features.

## Weeks 8 to 10

Complete initial coding of the more advanced feature (or features) chosen. *<This will depend on whether you have opted to develop a graphical user-interface, a complex sea-shell modeller, or a user-defined texture generator.>*

Milestone: A working but basic system not necessarily fully debugged.

## Weeks 11 and 12

Refine the code already written. Review remainder of project plan in view of program development to date and adjust as necessary. Write the Progress Report drawing attention to the code already written, incorporating some examples, and recording any augmentations which at this stage seem reasonably likely to be incorporated.

Milestones: Basic code now working, but probably with some serious inefficiencies, Progress Report submitted and entire project reviewed both personally and with Overseers.

## Weeks 13 to 19 (including Easter vacation)

Design and implement such augmentations as seem reasonable. Write initial chapters of the Dissertation.

*<The Easter break from lectures can provide a time to work on a substantial challenge (perhaps going beyond your initial plan) where an uninterrupted week can allow you to get to grips with a complex task. This is a good time to put in some quiet work (while your Supervisor is busy on other things) writing the Preparation and Implementation chapters of the Dissertation. By this stage the form of the final implementation should be sufficiently clear that most of that chapter can be written, even if the code is incomplete. Describing clearly what the code will do can often be a way of sharpening your own understanding of how to implement it.>*

Milestones: Preparation chapter of Dissertation complete, Implementation chapter at least half complete, code performs tolerably well and should be in a state that in the worst case it would satisfy the examiners with at most cosmetic adjustment.

## Weeks 20 to 26

*<Since your project is, by now, in fairly good shape there is a chance to use the immediate run-up to examinations to attend to small rationalisations and to implement things that are useful but fairly straightforward. It is generally not a good idea to drop all project work over the revision season; if you do, the code will feel amazingly unfamiliar when you return to it. Equally, first priority has to go to the examinations, so do not schedule anything too demanding on the project front here. The fact that the Implementation chapter of the Dissertation is in draft will mean that you should have a very clear view of the work that remains, and so can schedule it rationally.>*

Work on the project will be kept ticking over during this period but undoubtedly the Easter Term lectures and examination revision will take priority.

## Weeks 27 to 31

*<Getting back to work after the examinations and May Week calls for discipline. Setting a timetable can help stiffen your resolve!>*

Evaluation and testing. Finish off otherwise ragged parts of the code. Write the Introduction chapter and draft the Evaluation and Conclusions chapters of the Dissertation, complete the Implementation chapter.

Milestones: Examples and test cases run and results collected, Dissertation essentially complete, with large sections of it proof-read by Supervisor and possibly friends and/or Director of Studies.

## Weeks 32 and 33

Finish Dissertation, preparing diagrams for insertion. Review whole project, check the Dissertation, and spend a final few days on whatever is in greatest need of attention.

*<In many cases, once a Dissertation is complete (but not before) it will become clear where the biggest weakness in the entire work is. In some cases this will be that some feature of the code has not been completed or debugged, in other cases it will be that more sample output is needed to show the project's capabilities on larger test cases. In yet other cases it will be that the Dissertation is not as neatly laid out or well written as would be ideal. There is much to be said for reserving a small amount of time right at the end of the project (when your skills are most developed) to put in a short but intense burst of work to try to improve matters. Doing this when the Dissertation is already complete is good: you have a clearly limited amount of time to work, and if your efforts fail you still have something to hand in! If you succeed you may be able to replace that paragraph where you apologise for not getting feature X working into a brief note observing that you can indeed do X as well as all the other things you have talked about.>*

## Week 34

*<Aim to submit the dissertation at least a week before the deadline. Be ready to check whether you will be needed for a* viva voce *examination.>*

Milestone: Submission of Dissertation.