Generative Art 2002

Computation-Universal Voxel Automata as Material for Generative Design Education

Dr T. Fischer, MEd, PhD.
School of Design, The Hong Kong Polytechnic University.
e-mail: sdtom@polyu.edu.hk

Abstract

This paper is a report on the educational application of a voxel automata system for massively parallel execution of computation-universal cellular units in the generative design field. The software, designed and co-developed by the author to enable developmental strategies in generative design - for example with respect to 3D design generation, semantic self-evaluation and self-replication - was applied in teaching at the School of Design at The Hong Kong Polytechnic University to achieve two goals: to teach programming as part of the School's Interactive Systems Design stream and to teach generative concepts at a theoretical, yet hands-on and highly intensive level. An introduction to the software, its development and its functions as well as a discussion of the teaching/learning experience is given, highlighting design educational aspects and student design work. The paper concludes with an analysis of how student approaches to generative concepts have been affected by the tool and how ideas and feedback from students have supported the ongoing development of the voxel automata system and its documentation.

1. Generative Design as Interactive Systems Design

As part of the Hong Kong Polytechnic University School of Design's initiative in Interactive Systems Design, its 2nd year BA (Hons) design students have the opportunity to participate in a four-week project on Generative Design in the form of a profession-specific subject. These subjects, within the interdisciplinary BA (Hons) programme, are primarily intended to provide skill-based, technical training to complement the programme's cross-disciplinary, experience- and communication-focused elements. The teaching subject at issue (led by the author) provides skill-based training in basic programming techniques within the framework of a general introduction to Generative Design. This framework is used as an explorative context for programming experimentation. Together with the School's initiatives in the areas of design learning *by means of digital tools* (see for instance [6]) and design learning *about digital tools*, the Generative Design subject represents an example of the School's interest in communicating strategies of *designing (through) digital design tools* (see [7]). In this context, students are given the opportunity to study not only the application of design tools and methods but also the development of their own design tools and methods, understanding design as *toolmaking* (see [2]). Students are encouraged to consider computers not only as a *subject* of design, but also as *means* and *material* for design. Embracing skill-based technical training and conceptual generative design exploration, the teaching rationale of this subject is a strongly constructivist one, capitalising on student initiative, individual interest and prior

knowledge. The implicit aim of "generative" processes of comprehension was discussed in the context of Generative Learning theory [18] in [9].

The Generative Design subject described above has been taught twice so far in fall of 2001 and again in fall of 2002, and each time it was attended by four students. After focusing on techniques of Java 2D graphics generation in fall of 2001, the class discussed here was designed to allow for more spatial experimentation, using an easier and quicker-to-learn programming technology. This was achieved by applying the voxel automata system *Zellkalkül*, of which a detailed technical account is given in the following section.

2. Voxel Automata for Digital Morphogenesis

Originally developed as a framework for experimentation in digital morphogenesis research (see [7]), *Zellkalkül* is a stand-alone Java application with a graphical user interface to a boundless 3D voxel automata system. The system is based on a Cartesian coordinate system to address close-packed voxels, which can be displayed either as spheres or as rhombic dodecahedra (compare [17], p. 544 ff.).

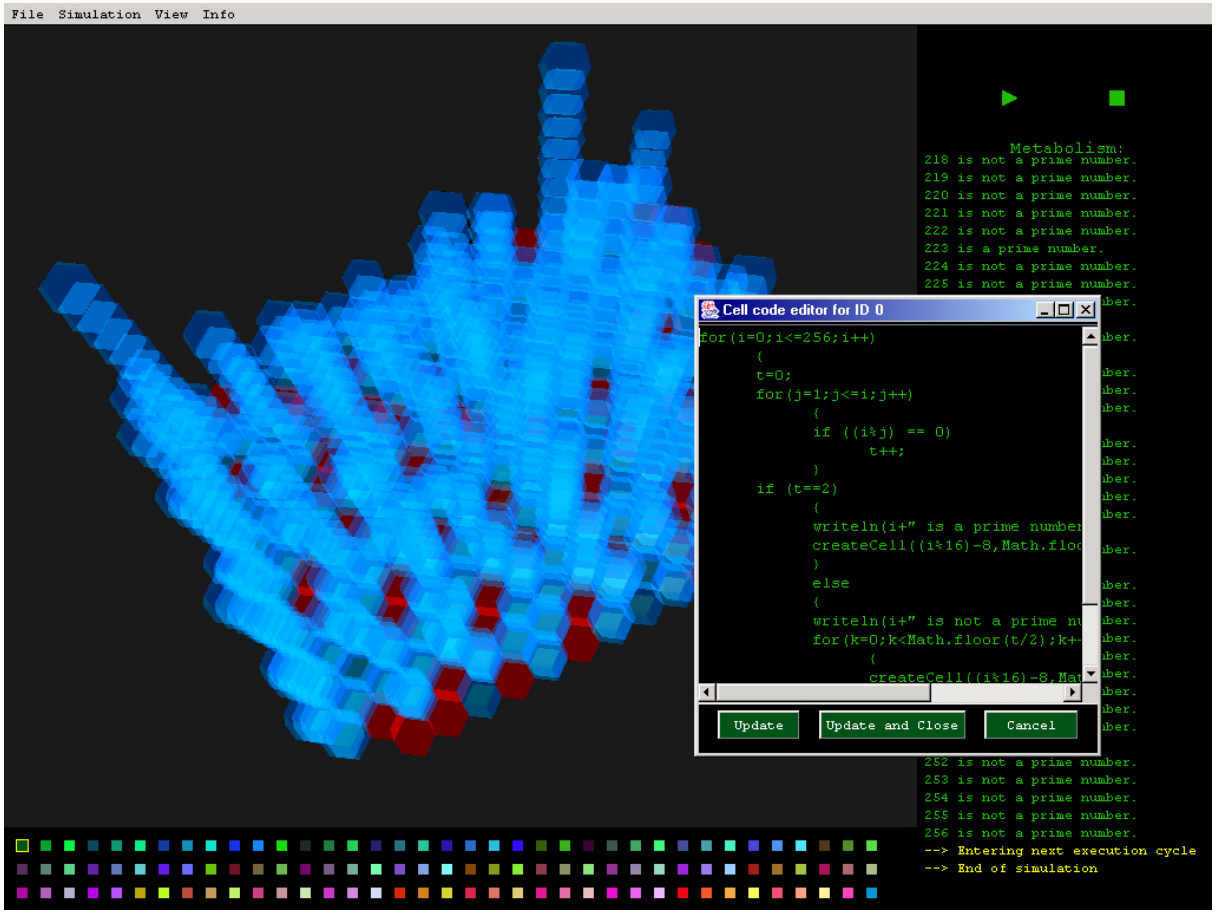

Figure 1: *Zellkalkül* user interface showing an example program/form

Zellkalkül's concept is based on principles of cellular automata and artificial life The three main differences between classic Cellular Automata systems and *Zellkalkül* are:

- *Zellkalkül* is three-dimensional and every cell has 12 neighbours.

- Its coordinate system is a boundless “universe” rather than a closed torus topology.

- Different cells in *Zellkalkül* can be governed by different behavioural patterns (”non-uniform automata”).

The behaviour of cells in *Zellkalkül* is not limited by simple rules. It can be of any degree of complexity since *Zellkalkül*’s cells are freely programmable. This environment allows textual programming of cells to generate cellular manifestations of formal and/or behavioural character. Structures generated in the system can be picked and moved, rotated and zoomed in and out using the mouse and the cell’s transparency can be globally controlled in ten gradients from highly transparent to opaque.

Voxels can have one of a number of identities (IDs), which are represented as coloured squares at the bottom of the graphical user interface shown in figure 1. Programming code can be associated with each ID. This code will define the behaviour of all cells that are instances of the respective ID during execution of the automata system. This allows the massively parallel execution of different code scripts in one automata system.

The programming language used is an extended version of ECMAScript [15], of which JavaScript and Flash ActionScript are descendants (Jean-Marc Lugrin's free FESI interpreter [14] was used). The language was extended by adding problem-centred elements for cellular morphogenesis including functions to create, inspect, manipulate and delete cells, as well as by provisions for intercellular communication and for parallel execution control. The functions allow basic operations (“create cell”, “delete cell”) as well as operations based on natural cellular paradigms (“split”, “die”). These language extensions, the voxel coordinate system and programming examples are covered in detail in a system handbook [8]. Both procedural and object-oriented programming are supported by the scripting language. The system is designed to support endless execution loops, like those that occur life-game type cellular automata. During each execution cycle, each cell can execute either its entire code or a defined (special-character-delimited) segment of it. The runtime model is also borrowed from life-game-type cellular automata systems. It does not terminate after all cells have executed their rule-based behaviours but keeps on looping indefinitely. However, cells can be explicitly excluded from and included in execution cycles with code respective functions. Once all cells are excluded from execution, a cellular program terminates.

As a tool for 3D shape and behaviour generation, *Zellkalkül* has similarities to, but also significant differences from other voxel software systems. Kai Strehlke has developed an online collaborative 3D modelling and shape morphing system named “xWorlds” and applied it in design teaching for example at the School of Design at The Hong Kong Polytechnic University (see [16]). XWorlds’ shape development is based on direct user manipulation and morphing techniques, and its parametric voxel-topology is based on cubic close-packing. In contrast to the three-dimensional educational (cubic) voxel modelling system DDDoolz [1] for example, *Zellkalkül* does not allow for manual (cursor-based) shape assembly. All form is generated by means of (multi)cellular programming. Compared to the three-dimensional environment for self-reproducing programmes described by Ebner [5], *Zellkalkül* is topologically based on close-packed spheres, each of which has twelve neighbours and it allows for higher-level cellular programming.

The program has undergone a number of different stages during its 10 months of (ongoing) development. It was inspired by observations made during the implementation and

application of a haptic programming environment (this aspect of the project background was discussed in [7]). But being also inspired by classic cellular automata systems, it bagan with a two-dimensional grid with all the topological features of a Game of Life system (upper left side in figure 2). At this stage, manual tissue composition was possible. Very soon, however, the limitations of a two-dimensional system became too dominant and a mock-up of a 3D alternative was created for discussion (upper right side in figure 2). Note that at this and all later stages shown in figure 2, a 3D-cursor and cursor-control buttons for manual cell assembly has still been envisioned. This approach was later abandoned in favour of purely code-driven manipulation. Following a suggestion by Prof. John Frazer, the cubical voxel arrangement was abandoned in favour of rhombic dodecahedra. Based on extensive previous experiments and applications with this topology (see for example [10], pp. 84 ff. and 98 ff.), Prof. Frazer's advice has helped in avoiding the problem of having two types of neighbouring relationships and distances between cubic voxels (side neighbours and point neighbours). It was also key to the author's (and successively the students') understanding of how cells in such a configuration could be easily addressed (see further below). The rhombic dodecahedron is a comparatively "natural" form and was identified and described as a common natural cell geometry as early as in 1815 (see [12]).

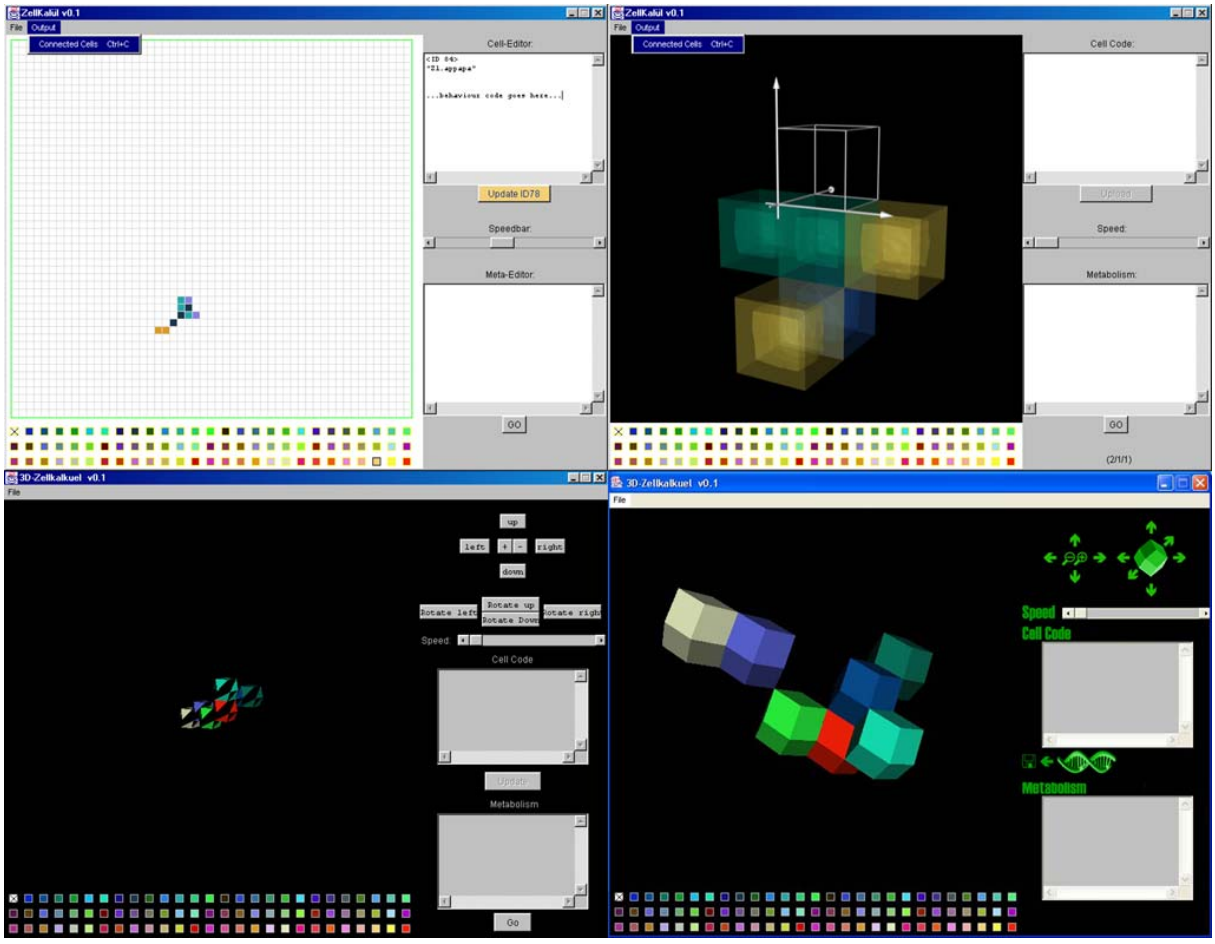

Figure 2: Evolution in *Zellkalkül* software development

The software represents an attempt to allow modelling of groups or colonies of parallel objects or particles (cells, atoms, molecules, planets, insects etc.) and nested systems of such groups in a very generic (application non-specific) manner. It is designed to potentially allow the simulation of any process of energy and material distribution/interaction in time and

space, based on temporal and spatial relationships as well as on internal (programmed) logic. In previous research applications, *Zellkalkül* has been used to simulate cellular development in biological organisms based on temporal and spatial cellular identity (see [7], pp. 118-121). In other research contexts it is currently being used to investigate the possibilities of applying natural principles of morphogenesis to man-made design and construction. *Zellkalkül*'s development is ongoing, and planned future extensions include networked computation support by server clusters and parametric control of cell geometries to enhance the degree of freedom in form generation. It is also planned to make use of these new features in future educational design applications.

3. Generative Design Learning

As mentioned above, this subject preceded a previous four-week Generative Design subject taught in 2001 (also led by the author) that made use of Java2D technology to generate different types of line drawings (variations on replacement systems/grammars, space-filling curves etc.). The 2001 learning and teaching experience suggested the use of an easier coding technology and more interesting design representations than two-dimensional line drawings in the 2002 follow-up subject. This led to the decision to apply the research tool *Zellkalkül* with its 3D view and ECMAScript coding technology to Generative Design teaching. This has, however, necessitated the introduction of a number of new, abstract and harder-to-understand concepts to the taught subject, such as massively parallel programming, morphogenesis based on cellular development and a variation of the Cartesian coordinate system that is neither commonly known nor easily understood. The following example illustrates how such abstract new subject matters were introduced in quick succession using concrete examples and tangible teaching materials.

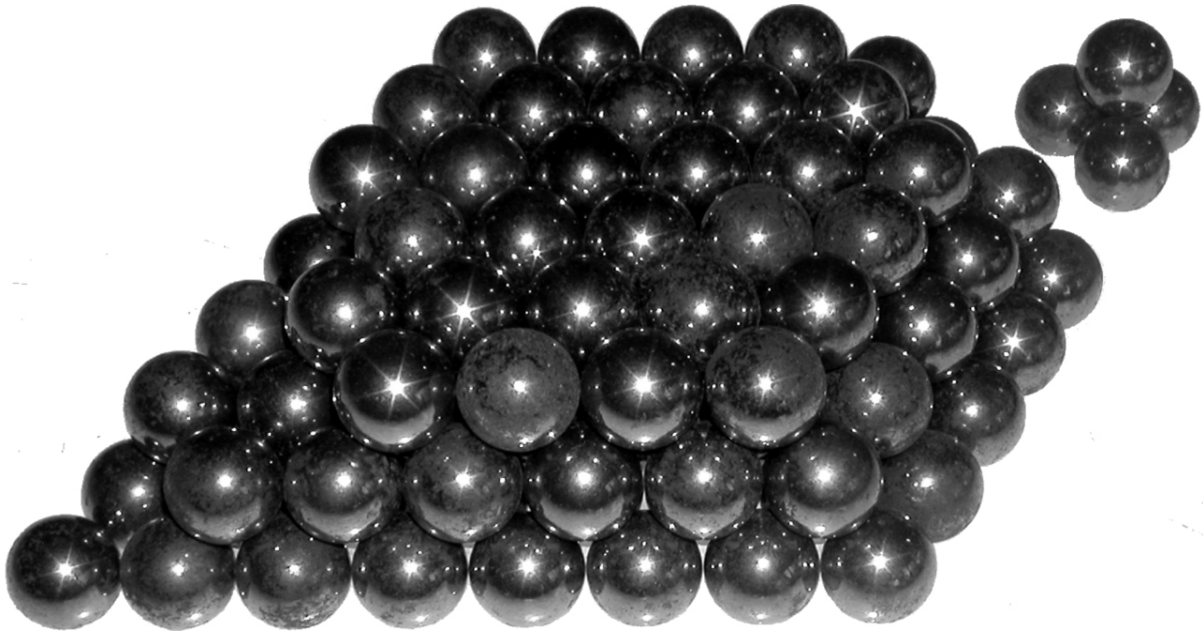

Figure 3: Physical models as learning aids: Ball bearing

Close-packed rhombic dodecahedra are arranged like spheres in hexagonal close packing (e.g. in face-centred cubic packing). Thompson gives an account on the relationship between spherical and rhombo-dodecahedral close packing on p. 552 in [17]. Figure 3 shows one of a

number of ball bearing boards used at the School of Design's Design Technology Research Centre (DTRC) in close-packing related projects. The boards were used in this teaching subject to allow hands-on close packing experiments. The experiments (see image above) show that, unlike cubic close packing, hexagonal close packing places every layer's elements above the gaps between the elements on the layer beneath it. Elements are hence not stacked in straight (cubic) lines in all three dimensions which suggests at first glance that they could not be addressed in a straight-forward way using natural numbers as x, y and z coordinates.

The rapid prototype model shown twice in figure 4 (also produced by the DTRC), however, shows, that such a mode of addressing is still possible. The model on the left shows a sphere with its twelve hexagonally close-packed neighbours. Rotating it 60° to the orientation shown on the right reveals that this arrangement has strong cubic characteristics. This rotation is the single difference between hexagonal and face-centred cubic close packing. After the rotation, on each of the three horizontal layers, all elements are arranged along straight lines in both dimensions on the horizontal plane. The deviation from cubic close packing is an offset that applies to all odd-numbered layers, which also has the result that the vertical distance between two layers is not the diameter of one element but the half of the square root of one element. This illustrates how each element can be addressed using natural numbers as x, y and z coordinates in a Cartesian coordinate system if the described offsets are taken into account when representing the structure. *Zellkalkül* uses a simple Cartesian coordinate system and addresses based on natural numbers in exactly this way. Though this possibility might not be immediately obvious, it could be explained and demonstrated to the students within a matter of minutes using these tangible models.

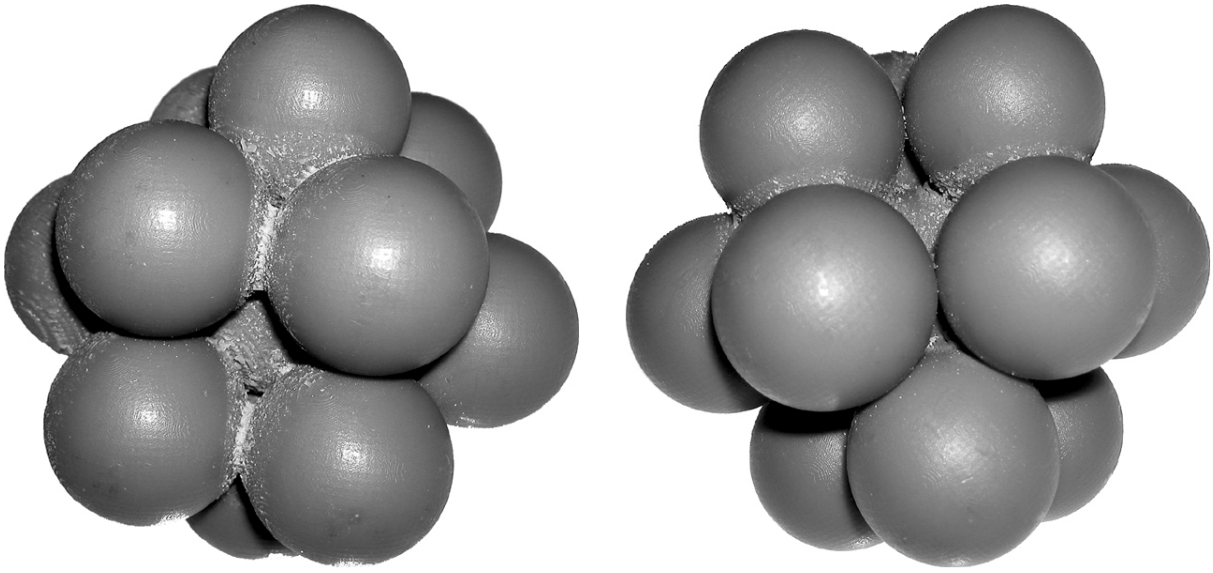

Figure 4: Physical models as learning aids: Rapid-prototype

From the author's and the students' perspective, it would have been highly desirable to be able to output structures generated in *Zellkalkül* on rapid prototyping machines. Unfortunately, while paying great attention to 3D data import, the Java(3D) development community has not yet provided a self-contained free solution for outputting Java3D data (*Zellkalkül*'s internal data format) as common 3D data formats such as .3ds or .stl. At this writing, an attempt to develop this possibility for *Zellkalkül* is being undertaken but could not be offered to the students of this subject.

As stated above, a code interpreter is associated with each of the spherical or rhombo-dodecahedral elements. Since the cell scripting language used is closely related to popular Web scripting languages such as JavaScript, JScript and Flash ActionScript, students were enabled to reapply their learned programming experience in later interactive Web design projects. To facilitate online code sharing and discussion, a "project hub" website was put online containing links to the *Zellkalkül* Handbook [8], Email links to all class participants and links to all project/data sharing folders of all class participants. These project and data sharing folders were set up on a departmental data storage and exchange server cluster for design teaching to which all students and staff have access (see [3]). The project hub website itself was used as a demonstration of how learned scripting knowledge could be reapplied in on-line authoring. It was based on JavaScript-based dynamic HTML, generated by code elements previously discussed in this subject. An analysis of the website's source code was used to re-iterate previously acquired programming knowledge.

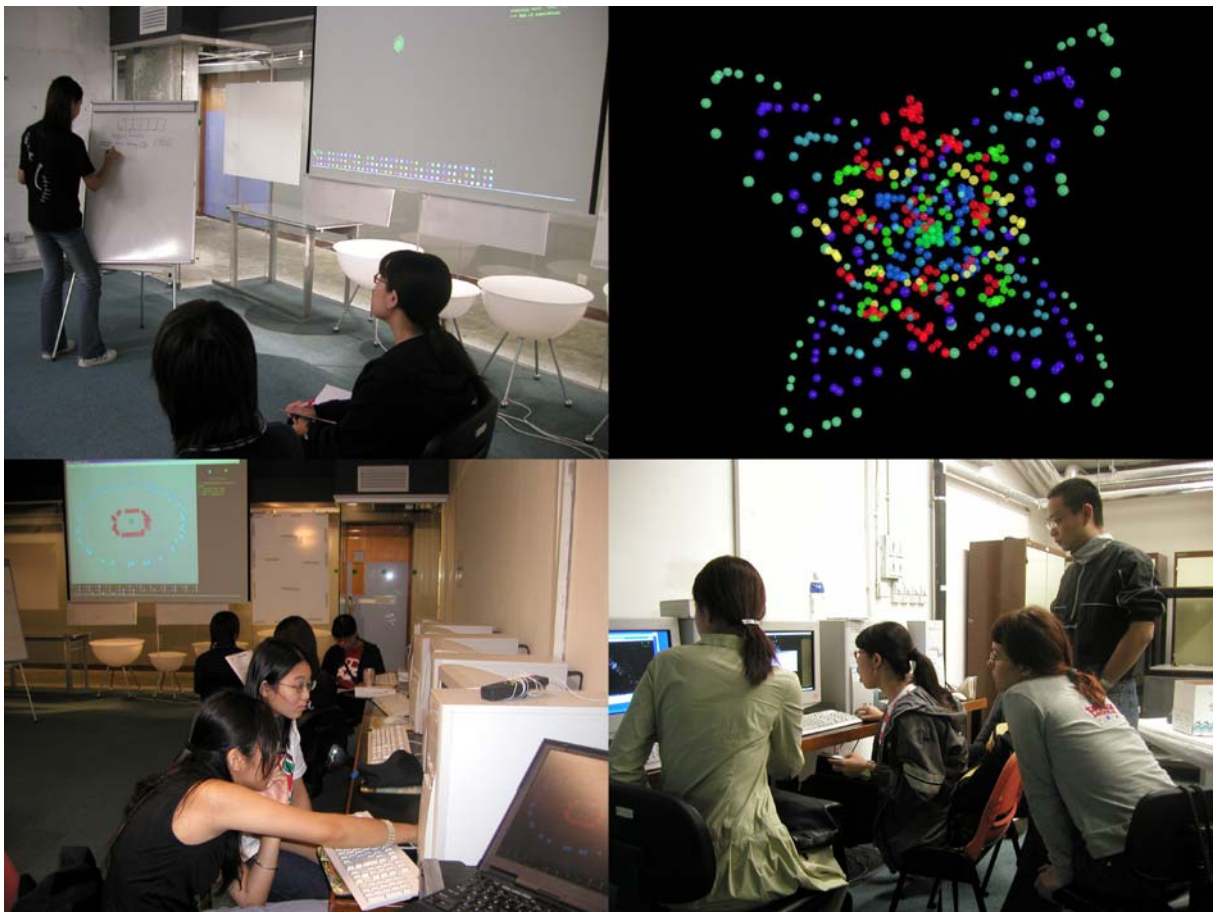

Figure 5: *Zellkalkül* coding exercises at the School of Design

In order to engage students in programming learning at a very high degree of activity and also to allow them to quickly achieve confidence in software development terminology, students were asked to research, prepare and deliver short presentations on programming language elements and to discuss related examples (variables and arrays, functions, objects, different loop types, conditional branching, etc.). Consequently, individual students became "experts" in particular code constructs and data structures, which was then used to build up a mutual programming support system amongst the group of students.

Since this subject focused on skill (programming) learning rather than on the production of design output of some kind, emphasis was put on the student's learning and thinking processes, their interaction and mutual support, which had to be closely observed and analysed. In order to allow such observation, the School of Design makes wide use of the "lablog" concept. Barbara Dass and John Frazer coined the term "lablog" in 1991. It refers to a combination of a laboratory notebook and a ship's logbook of a journey of discovery (see [4], p. 2) and describes a very free format for creating learning process/progress protocols. At the School of Design, lablogs are most frequently produced in form of A5, A4 or A3-sized notebooks or bound paper collections containing notes in various formats such as text, images, drawings, polaroid pictures, origami experiments, news clippings and so forth. These documents are typically reviewed together with students in tutorials during a subject and collected and analysed at the end of a subject for assessment purposes. Figure 6 shows an example page of one of the lablogs produced in this Generative Design subject: a reflection on how loop structures and trigonometric functions can be used to create elliptical and line structures with parametric variations based on the day (numerical date) of program execution.

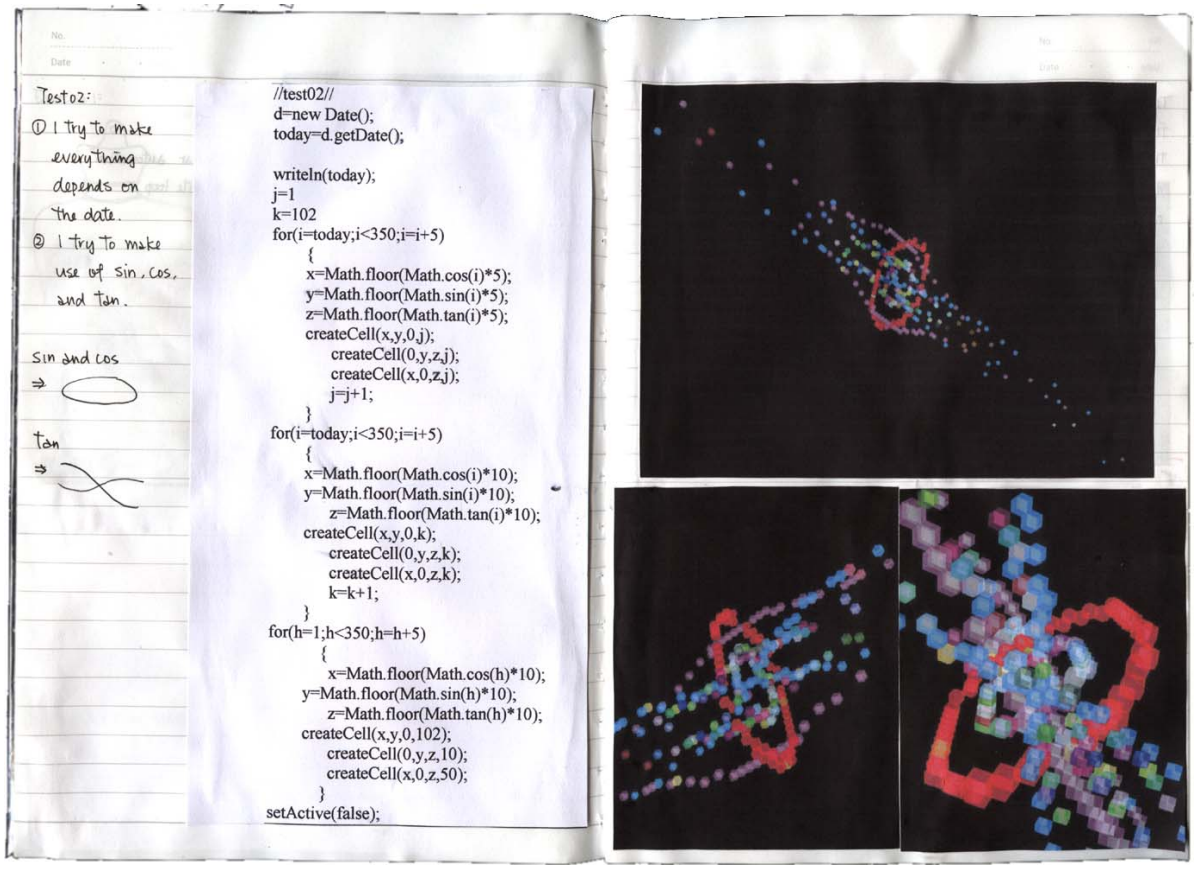

Figure 6: "Lablog" showing programming experimentation

In this subject, generative systems were represented as verbally described ideas, programming code, written text, sketches, virtual models of different types, screen shots and collaged lablog reflections. This multi-modal representation of concepts in this subject has led to a very intense learning experience.

Surprisingly, though the software used allows the generation of any kind of cellular pattern and though natural tissues typically consist of solidly close-packed cells, the students in this subject showed great interest in very loose, line and surface-type configurations. One reason

for this might be that, despite adjustable cell transparency, it is rather difficult to visually understand the inner structure of solid configurations, which is made more difficult by the large number of faces at different angles in different layers when viewing cells as rhombic dodecahedra. By generating loose and rather sparse structures, students achieved configurations that were very easily comprehensible on the two-dimensional screen by moving and rotating them. Accordingly, students developed a strong interest in line, curve and circle drawing algorithms rather than in control structures that are more suitable to fill space solidly. Before this subject the students had little or no previous programming experience. Because of this lack of preconception, it was expected that they would unquestioningly embrace the massively parallel programming paradigm. Nevertheless, they tended strongly towards structures developing from code in one single cell and terminating this program after the first execution cycle instead of programming multiple cells using intercellular code manipulation and communication or repeating execution cycles. One reason for this might be the simplicity and sequential linearity of most programming examples in the *Zellkalkül* handbook at that time. The software, many of its code functions and the handbook were further developed over the course of the subject to reflect emerging problems and ongoing discussions. The mentioned tendency of the students led to an emphasis on multicellular programming and runtime coordination in successive additions to the handbook, and to the incorporation of more code functions for runtime management. However, this did not change the students' bias towards procedural coding executed as single-cell programmes.

4. Conclusion

The teaching subject discussed above demonstrates a successful application of research tools to learning and teaching. It succeeded not only in making efficient multiple use of a software development project but also to make students aware of some present-day design research approaches and tools. Up to this point, experimentation in *Zellkalkül* has tended towards abstract and rather sketchy results. Since the discussed subject was more process-centred than product centred, this was not seen as a problem and it has helped to avoid overly iconic "3D drawing". Understanding the software and in particular its coordinate system poses some challenges for learning and teaching especially in a very short subject. But using tangible learning materials and encouraging mutual student support have proven to be very effective in this context. In general, this subject and its outcomes have been regarded as more interesting, and its learning has been experienced as more effective and worthwhile than in the previous Generative Design subject in 2001 using Java2D. In this subject, mainly due to time constraints, only a small fraction of the versatility and power of *Zellkalkül* was made use of by the students. It became obvious that this research tool could serve as a learning environment for much longer and more intense teaching subjects and experiments in the Generative Design field. It is planned to expose students to *Zellkalkül* for longer durations in the future. The use of the programme by users without previous programming knowledge, who can take a fresh view at it, is expected to continue to drive the further development of this research tool, its applications and its documentation.

5. Acknowledgements

I gratefully acknowledge the valuable support from my colleagues at the School of Design at The Hong Kong Polytechnic University and at the Spatial Information Architecture

Laboratory at the Royal Melbourne Institute of Technology University, in particular Prof. John Frazer, Prof. Mark Burry and Timothy Jachna whose feedback has been vital to this teaching project. I also acknowledge the excellent teaching support by Nicole Schadewitz and the superb software development support by Torben Fischer, without which the discussed software could not have been implemented. Special thanks also to the 2002 students of the ISD PSS sd3900: Huang Ying Grace, Lau Mei Ki Miki, Wong Kwan Fong Kim and Choi Wai Bun Gary. This project is supported by Departmental General Research Funds at the School of Design at The Hong Kong Polytechnic University (Project Code A-PB84).

5. References

[1] Achten, H.H., Vries, B. de and Jessurun, J. (2000). DDDoolz – A Virtual Reality Sketchtool for Early Design. In: Tan, B.K., Tan, M. and Wong, Y.C. (eds.), CAADRIA 2000: *Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia*, National University of Singapore, Singapore, pp. 451-460.

[2] Ceccato, C. (1999): Microgenesis. The Architect as Toolmaker: Computer-Based Generative Design tools and Methods. In: Soddu, Celestino (ed.): *The Proceedings of the First International Generative Art Conference*. Generative Design Lab at DiAP, Politechnico di Milano University.

[3] Ceccato, C., Fischer, T., Li, C.M. and Frazer, J. (2002): A Large-Scale Computing Infrastructure for Design Education. In: Koszewski, K. and Wrona, S. (eds.): *Design e-ducation. Connecting the Real and the Virtual. Proceedings of the 20th eCAADe Conference*. Faculty of Architecture, Warsaw University of Technology, Warsaw 2002, pp. 282-289.

[4] Dass, B and Frazer, J. H. (1991): *DES101 Design Thinking. Course Handbook*. Department of Design in Industry, Faculty of Art and Design, University of Ulster.

[5] Ebner, M. (2001): A Three-Dimensional Environment for Self-Reproducing Programs. In: *Advances in Artificial Life: 6th European Conference, ECAL 2001*, Prague, Czech Republic, Berlin and New York: Springer, pp. 306-315.

[6] Falk, L., Ceccato, C., Hu, C., Wong, P., and Fischer, T. (2000): Towards a Networked Education in Design. A First Manifestation through the "Virtual Design Company" Studio, In: Tan, B.K., Tan, M. and Wong, Y.C. (eds.), CAADRIA 2000: *Proceedings of the Fifth Conference on Computer Aided Architectural Design Research in Asia*, National University of Singapore, Singapore, pp. 157 - 167.

[7] Fischer, T., Fischer, T. and Cristiano C. (2002): Distributed Agents for Morphologic and Behavioral Expression in Cellular Design Systems. In: Proctor, George (ed.): *Thresholds. Proceedings of the 2002 ACADIA Conference*, College of Environmental Design, California State Polytechnic University, Pomona, Loa Angeles, 2002, pp. 113-123.

[8] Fischer, T. (2002): *Zellkalkül. User's Handbook*. Unpublished.

[9] Fischer, T. and Herr, C. M. (2001): Teaching Generative Design. In: Soddu, C. (ed.) *The Proceedings of the Fourth International Conference on Generative Art 2001*. Milan, Italy: Generative Design Lab, DiAP, Politechnico di Milano University.

[10] Frazer J. H. (1995). *An Evolutionary Architecture*. London: The Architectural Association.

[11] Gerhart, J. and Kirschner M. (1997). *Cells, Embryos, and Evolution. Toward a Cellular and Developmental Understanding of Phenotypic Variation and Evolutionary Adaptability*. Malden, MA, Blackwell Science.

[12] Kieser, D. G. (1815). *Phytotomie, oder Grundzüge der Anatomie der Pflanzen*. Jena, Cröcker.

[13] Kolarevic, B. (2000). Digital Morphogenesis and Computational Architectures, *SIGraDi'2000 - Construindo (n)o espacio digital (constructing the digital Space)*, 4th SIGRADI Conference Proceedings, Rio de Janeiro (Brazil) 25-28 September 2000, pp. 98-103.

[14] Lugrin, J. M. (2000): FESI ECMAScript Interpreter v1.1.5. URL: http://home.worldcom.ch/˜jmlugrin/fesi/, 2000. On 05-Nov-2002.

[15] McComb, G. (1998). ECMAScript Language Specification. Lincoln, iUniverse.

[16] Strehlke, K. (1999): xWORLDS. The implementation of a three-dimensional collaborative sketch tool within the context of a third year design course. In: Soddu, Celestino (ed.): *The Proceedings of the First International Generative Art Conference*. Generative Design Lab at DiAP, Politechnico di Milano University.

[17] Thompson, D. W. (1992). *On Growth and Form. The Complete Revised Edition*. Dover Publications, New York.

[18] Wittrock, M. C. (1990). Generative Processes of Comprehension. *Educational Psychologist*, 24(4), pp. 345-376.