Advanced computing with cellular automata

by Andy Yang, Jamie Hirst, Pedro de Souza, Herah Khan and Andrew Shaw

Software development in today's competitive world is about more than just writing ("coding") a computer program. Software development is, however, a complex process that involves the capturing of detailed specifications and documentation, cooperating efficiently in task groups, and consulting with the client, thereby gaining an understanding of what the client expects, and fulfilling the client's requirements with a stable, high-quality product.

Students in the University of Pretoria's Department of Computer Science do a third-year module on software engineering, which gives them an opportunity to learn about all those skills and many more. This course also requires students to complete a practical year project, which incorporates and assesses the software development skills. One of the students' projects involved the design and programming of a cellular automata simulator.

Cellular automata

A cellular automaton (CA) is a regular grid of cells that form a "world". The grid has finite dimensions and each of its cells has its own internal state.

The project group was given the opportunity to work on a software simulator for cellular automata applications. The system made provision for triangular, rectangular and hexagonal worlds. Their software allowed the user to easily enter the local transformation rules for such worlds, to accurately simulate the desired phenomena, and to visualise the worlds in either two-dimensional or three-dimensional graphics, thus bringing a simulation to "life" in an aesthetically astonishing manner.

Internal states

Cellular automaton cells represent a world and each has a particular internal state. In a similar way, the cells in a cellular automaton also have internal states that can change over time while the automaton is active. In their software system, the students used integer numbers to represent those internal states. This provided the user with a very large number of possible cell states for world simulations, and allowed for very complex and feature-rich worlds in their cellular automata simulation environment.

Value ranges

The simulation of a cellular automaton on "raw data" can often look complex and confusing. However, by allowing the user to associate intuitive colours with ranges of internal state values, the software makes such simulations more meaningful, understandable, sensible and comprehensive to the user. In the students' software system, all value ranges are entered in a manner that is familiar to mathematics students, for example, $2<x<7$ is entered as (2;7), whereas $2<=x<7$ is entered as [2;7). The software system even supports the negative infinity value, which is represented as NEGINF, as well as the positive infinity value, which is represented as POSINF.

In order to assist the user in choosing a colour for a specific numeric range of states, the students programmed a panel, shown in Figure 1, which allows the user to slide the bars when changing the RGB colour values. Thus, the user is able to easily select the appropriate colour he or she desires, and the program automatically converts the RGB values to the popular hash colour schemes used in HTML when the user saves the chosen settings.

Transformation rules

The beauty of a cellular automaton and its ability to dynamically model so many different phenomena lie in its transition rule system. This rule system consists of a finite set of rather simple rules that are applied at many places simultaneously to advance a cellular world to its next configuration. In this way, one can evolve a cellular world.

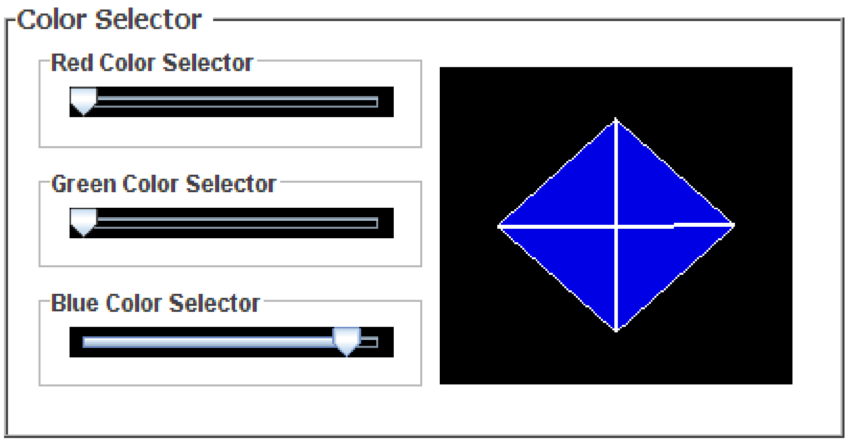
A rule is typically generic in the sense that it applies to a set of cells, not just to one particular cell, and it consists of three main parts: the current values/states of a cell's neighbours (application precondition), the current value/state of the cell itself (application precondition) and the value/state the cell will change to, should the two previous requirements be met for a rule (application postcondition). Hence, a rule will only be applied if the first two requirements hold, and the cell that matches the two requirements will be "evolved" to the new state, which is specified in the application postcondition mentioned above (everywhere, and simultaneously, in the entire cellular automaton).

In many situations, only certain values or states are of importance for a rule. To allow a rule to ignore the values or states of neighbours that are not of importance, the group made provision for a special symbol in the rule entering, namely the "*", which has the meaning of "any" or "accept all" (see Figure 2). It enables the user to focus only on those cell states or values that really matter (rather than the ones that do not). This "any" symbol, which was adopted from the theory of typed graph transformation systems and generalised cellular automata, allows the values or states specified in the first two application preconditions mentioned above.

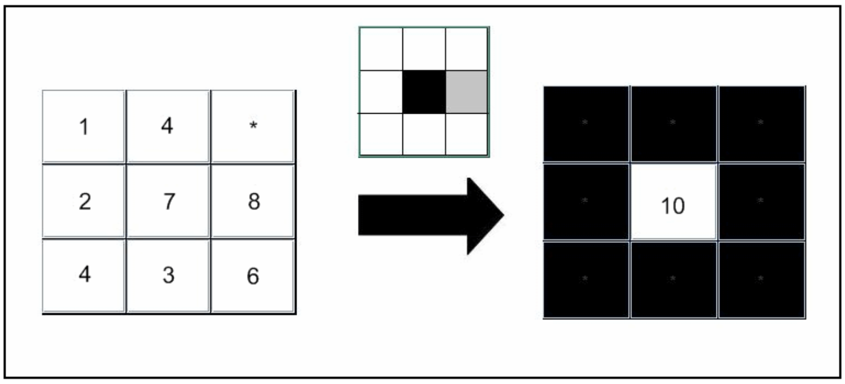
Multiprocessing

Multiprocessing is one of the most useful and complex capabilities of the latest computers. It can make a program far more efficient and helps to increase the speed at which a simulation takes place. Consequently, the students decided to implement their software system in such a way that it is suitable for multiprocessing.

Their software allows a user to choose between one, two, four and eight processors (CPUs). Consequently, it divides the rows (of a world) into the same number of segments and creates a processing thread for each segment. Each thread is then responsible for displaying and "evolving" the segment of cells allocated to it (see Figure 3). When this feature is utilised, the world in the students' simulations can evolve at a reasonably rapid pace.



1. RGB changer.



2. Rule-entering panel.

Simulation

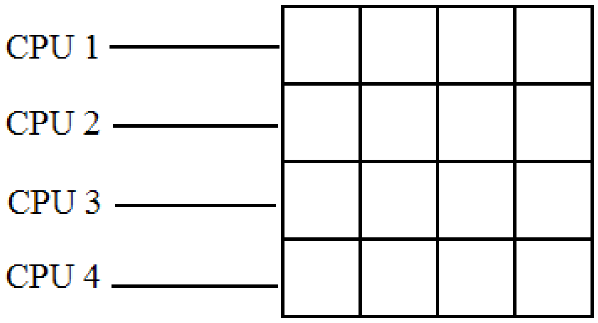
The graphical display of any software-based simulation is very important to the user, as it influences what the user can learn from performing this simulation. It is often desirable to view a simulation in a specific manner, from a particular perspective, height, angle, colour scheme or distance. The software tool developed by the students caters for this by allowing a user two-dimensional (2-D) and three-dimensional (3-D) modes.

Two-dimensional mode

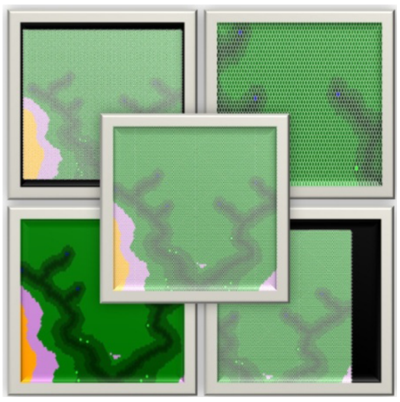
This interface mode allows the user to select multiple cells and easily adjust their values. Some phenomena are easier to view in a 2-D manner and such functionality is also provided in this view. The software allows users to edit and view the simulation however they want: zoom (in and out) and translate (up, down, left and right). See Figure 4.

Three-dimensional mode

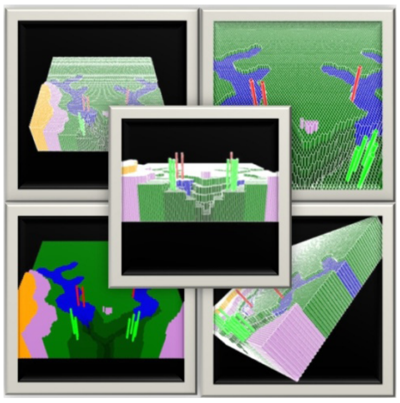
This interface mode is designed to run a simulation. It allows the user to rotate a world (up, down, left and right), zoom (in and out) and enables scaling, thus allowing the user to easily find the perfect way to view his or her simulation. See Figure 5.



3. Division of world into four segments for four CPUs.



4. Two-dimensional viewing feature.



5. Three-dimensional viewing feature.

Practical applications

When people think of cellular automata, they often think of Conway's *Game of Life*. This relates to physical phenomena such as algae growth or the spreading of a virus. However, since cellular automata are known to be Turing-complete, their applicability is constrained only by the theoretical limits of Turing-computability. Cellular automata can be thought of as an advanced form of a programming language, whereby the applications for which it can be used are mainly limited by the capability of the programmer to express his or her ideas.

Some interesting applications are possible in computer graphics nowadays. These are especially apparent in the field of computer games, where images need to be rendered at an efficient speed. In this particular field, a large amount of program code is usually required to determine the colours of particular pixels (in terms of textures, normals and lighting). Imagine being able to program an entire world while only worrying about a few simple sections and leaving the rest to the system. This will drastically reduce the amount of program code to be written by the programmer, whereby many mistakes will be avoided, as the underlying "engine" handles the rest, for example, choosing the correct materials, lighting and perhaps even the textures of a graphical scenario. See Figure 6.

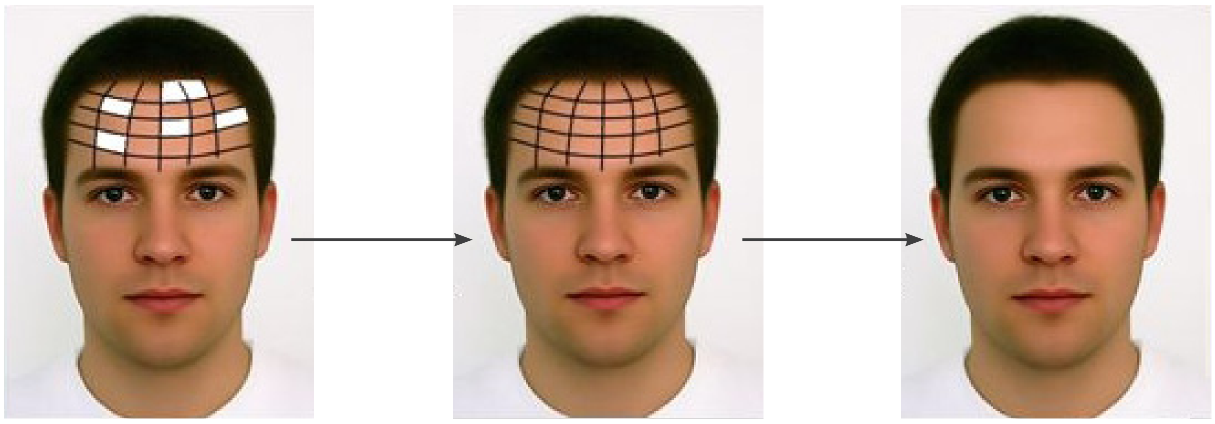
In the field of security, cellular automata can offer biometric authentication in an easy and simple manner. This can be done, for example, by dividing the eye or fingerprint into a grid of cells (world) and determining the relationship of each cell to its neighbours. A neural network could also be trained in this manner, through supervised learning. Afterwards, it could be used for a strong form of authentication. See Figure 7.

The project group's aim was to show how powerful and diversely usable a cellular automaton can be by illustrating this in a manner that non-experts would also understand and be able to appreciate. In order to do so, three main simulations were developed.

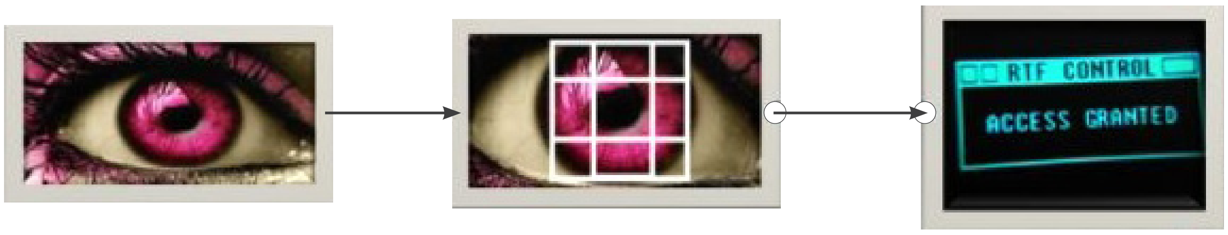
During the FIFA 2010 World Cup, many companies were looking for new and innovative ways to reach as many tourists as possible. To this end, the student project team presented a cellular automaton application, which offered any organisation the visual facility to reach new users or viewers in a new, intriguing and exciting manner. This would not only draw attention, but also have a lasting effect on all who see it. Think of a 3-D image in the middle of the football stadium that represents the flags of the countries of the teams that are playing, or a banner that forms new patterns for as long as one looks at it (see Figure 8).

In South Africa, many accommodation facilities are built sporadically, without much preceding research about their location. This is usually the case in rural areas and is often due to a lack of funding for such building projects. The aim of the group here was to show how a cellular automaton application could be used to analyse flood problems in landscapes (see Figure 9).

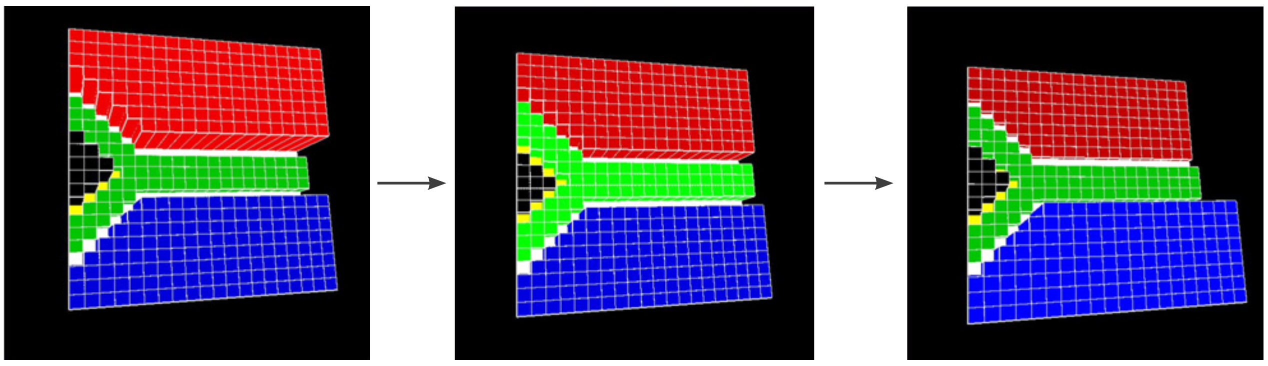
In many secondary schools in South Africa, learners are not sufficiently interested in pursuing scientific or computer science-related studies, perhaps due to their perceived difficulty of these subjects or a lack of prior exposure.



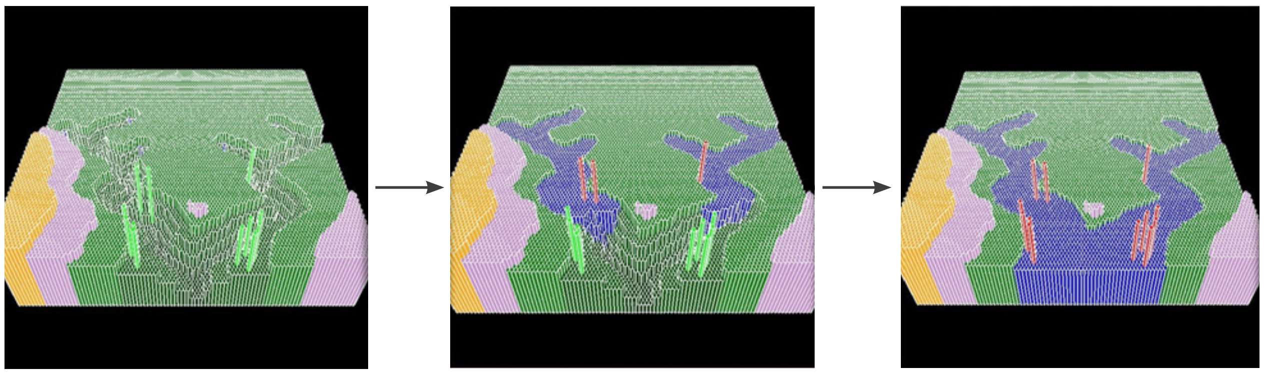
→ *6. Graphics simulation potential.*



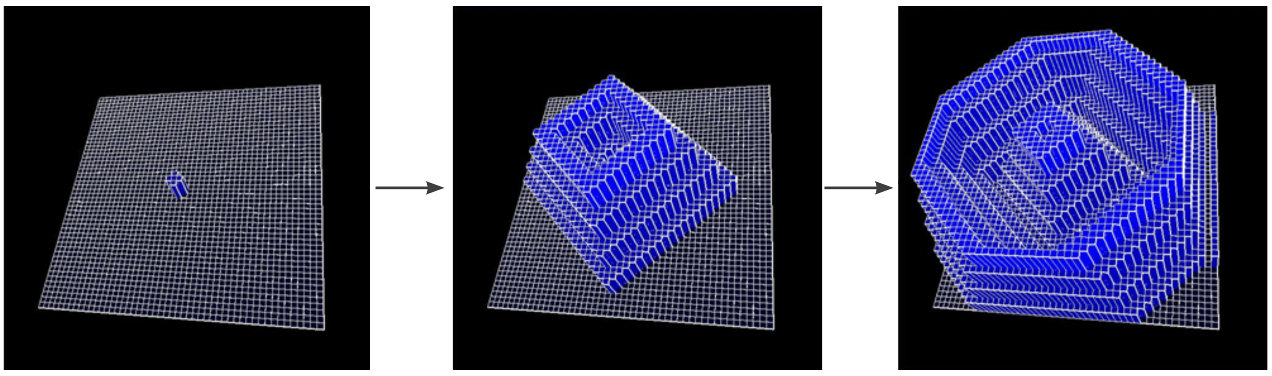
→ *7. Biometric authentication demonstration.*



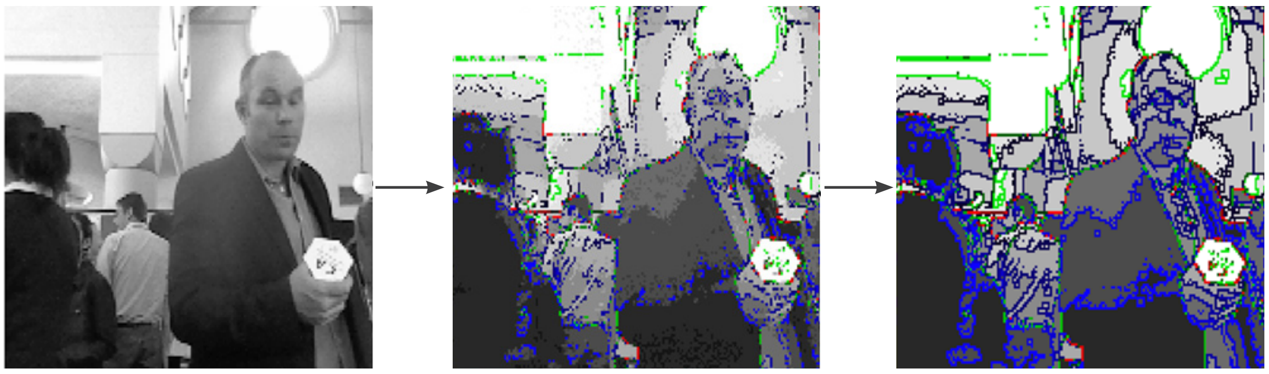
→ *8. Dynamic South African flag simulation (swaying).*



→ *9. Flood plane simulation.*



10. Simulation of a ripple effect on a liquid surface.



11. Edge detection simulation.

In this context, an intuitive cellular automaton visualisation can be used for educational purposes. The students designed a simulation of a ripple effect on a liquid surface (see Figure 10). This simulation allows a user to choose a place to make a drop and to see the wave impact it would have on the liquid surface. Similar simulations can be designed to show chemical bonding, chemical reactions or other phenomena that learners find hard to understand.

Another simulation is capable of discovering edges in digital images. This can be used for feature analysis, artificial intelligence and biometric authentication. For this cellular automaton application, the students developed a cellular automaton program that takes a photograph from a digital camera and converts it into a cellular automaton grid (world) (see Figure 11).

Acknowledgements

The student group would like to thank Morkel Theunissen for teaching them about design and development of complex software systems, Rushka Venter for all her help throughout the year, Rudi Penzhorn for his continuous guidance, support and assistance throughout this project, Retief Fourie for some additional interesting ideas, and last, but not least, Stefan Gruner for his scientific advice about the theory of cellular automata.

References

1. Gruner, S. 2009. Mobile Agent Systems and Cellular Automata. *Journ. Autonomous Agents and Multi-Agent Systems*, Vol. 20, pp. 198-233, Springer-Verlag.
2. Hey, A. (Ed.). 2002. *Feynman and Computation – Exploring the Limits of Computers*. Westview Press.

The following students were involved in this project: ***Andy Yang****,* ***Jamie Hirst****,* ***Pedro de Souza****,* ***Herah Khan*** *and* ***Andrew Shaw****. Their software system won two prizes: the best engineered software engineering project of the year at third-year level for software engineering methodology and the best overall project of the year for software product quality.*