Teaching Parallelism With Gamification in Cellular Automaton Environments

Antonio J. Tomeu Hardasmal and Alberto G. Salguero

Abstract—**Parallel programming within the computer science degree is now mandatory. New hardware platforms, with multiple cores and the execution of concurrent threads, require it. Despite the above, the teaching of parallelism with the usual methods and classical algorithms, make this topic hard for our students to understand. On the other hand, teaching complex topics through the techniques of gamification has already demonstrated, in a reliable way, a positive reinforcement of the student in front of the learning of complex concepts. In this work we demonstrate a way to convey the teaching of parallelism to undergraduate students using gamification in microworlds. The results obtained by the students who followed this model, compared to a control group that followed the standard model, show a statistically significant advantage in favor of the teaching of parallelism, using a gamification with microworlds model.**

Index Terms—**Cellular automaton, E-learning, gamification, microworlds, multicore programming, parallel programming.**

I. Introduction

THE teaching of concurrent and parallel programming has become, for a few years now, one of the cornerstones of the training of professionals in information and communication technologies (ICTs). According to estimates by the European Commission [9], between now and 2020 more than half a million specialists in these areas will be needed to meet the demand forecast by industry. Consequently, ensuring that the training of these professionals meets the deadlines set in their respective degrees, thus optimizing the graduation rate seems a necessity. However, this is far from being the case, and the difference between the number of years required to obtain the degree and the number of years required to do so is, unfortunately, very large. Currently, in many European countries, including Spain, the level of knowledge of science, and more specifically of mathematics, has decreased drastically, if we compare them with Asian countries such as South Korea or Japan. This leads to undergraduate students with little ability to abstract, to use mathematical language, and to algorithmically systematize a problem.

With this state of the art, it is not strange the learning of programming in general [31] and of parallel programming in particular, has become an authentic *tour de force* for our students [19], and that the success and performance rates of the courses where they are taught are relatively low. The examples and case studies that teachers use to convey our course matter are also unhelpful; although they are formative and affordable, they are generally very abstract, and do not stimulate students to program them, nor do they once programmed leave them with feelings of reward for tackling new developments.

In order to try to solve the problems exposed, during the academic year 2017/2018, we have developed a teaching experience that has taught students to program in parallel through the use of ludification, using microworlds. Instead of the classical case studies, we have chosen others based on microworlds that, in first place, stimulated students to program them and in second one, once programmed, provided them with a feeling of reward effective enough to tackle new developments with interest, increasing the commitment of the students. The objectives pursued with the experience have been:

- A: to measure the impact of gamification on students' performance in parallel programming.
- B: to measure the degree of effort of students with the model proposed in the experience, as opposed to the model of habitual use.
- C: to measure whether ludification facilitates the learning of multicore parallel programming.

The experience was developed with four concrete experiments represented through microworlds, and the students developed with them the practices of our course of "Introduction to Parallel Programming" during the academic year 2017-2018. The control group used was the group of students from the previous academic year, who developed the standard practices, to measure the impact of the model proposed by the experience. Subsequently, the results obtained by the students of both groups were compared by means of objective and subjective measures, in order to contrast if the proposed objectives had been reached. The results obtained leave no room for doubt, and show statistically significant evidence in favour of teaching multicore parallel programming by means of ludification with microworlds.

II. Background

Ludification originates in the former Soviet Union as an alternative incentive to remuneration for doing a job, and for some years has been configured as a novel strategy in academic practice [2], [3], [6], [7], [12]–[14], [18], [24], [28]. Its practical implementation admits two variants: the first is

Manuscript received October 16, 2019; revised January 6, 2020; accepted January 20, 2020. Date of publication March 6, 2020; date of current version April 6, 2020. (Spanish version received March 12, 2019; revised May 21, 2019; accepted September 24, 2019.) *(Corresponding author: Antonio J. Tomeu Hardasmal.)*

The authors are with the Department of Computer Science, University of Cádiz, 11003 Cádiz, Spain (e-mail: antonio.tomeu@uca.es).

There exists a Spanish version of this article available at http://rita.det.uvigo.es/VAEPRITA/V8N1/A5.pdf

Digital Object Identifier 10.1109/RITA.2020.2979067

1932-8540 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

staging, which assigns roles to students (usually related to their future professional practice) and makes them interact with each other; the second variant uses games where the student interacts with them and learns something, or where the students implement the game based on given rules, being the implementation process the one that leads them to learn that something. It is this last alternative, of which there are some recent precedents in literature for teaching programming [4], [10], [11], [15], [17], [20], [22], [23], [25]–[27], [31], that we have chosen, since it is uniquely well suited to teaching parallel programming. The use of ludification in the teaching of programming has always sought to reinforce certain aspects of programming learning, or to promote the active participation of students. Being a relatively new topic, there are nevertheless studies [6] that provide evidence of changes in student behavior. The concept of micro-world has been developed in the field of Artificial Intelligence and has been defined as "computer-based interactive learning environments, where prerequisites are incorporated into the system and where students become active architects of their own learning". The use of computers facilitates the analysis of simulated environments, the implementation of strategies, and also introduces a component of immediacy and interactivity that are uniquely useful for stimulating students [8, 19].

Notwithstanding the above, there are other elements to be taken into account, whose interaction produces synergies that facilitate the learning process, and which are the following:

- Students: is the subject who learns, in our case, parallel programming.
- Context: defines the set of activities defined by the teacher for a given microworld, and which oblige students to plan their implementation.
- Technical: defines the set of programming language tools that students must to use to support their implementation of the microworld, in our case oriented to multicore parallel programming.

The use of simulated micro-worlds has become widespread in a multitude of educational settings, including higher education [14], [15], and even in company training and education plans. By reproducing real-world situations, it aims at a user-platform interaction where decision making leads to successful or failed ends, offering the student feedback information about their interaction [2] from which to draw useful conclusions in the future. In fact, there are in the market a great variety of platforms of specific purpose that allow to learn programming by means of ludification to adolescents (Scratch, developed in the MIT), neurosciences to students of medicine (Future-Learning-PFL) or dynamic geometry and students of physics and mathematics (Cabri-Géométre). On the other hand, and from a more playful perspective, it is well known that any adolescent or young adult currently spends much of their time interacting with video games [32]. The immediacy of the reward, together with the interaction with the -sometimes very complex- virtual worlds they recreate, make them enormously attractive to them, since they obtain almost instantaneous positive feedback.

III. MICROWORDLS WITH CELLULAR AUTOMATON

There are multiple definitions of the concept of cellular automaton in the literature. In many fields they have been used as microworlds to model physical realities of high complexity; the propagation of forest fires, the percolation of substances, the combination of solutes of a chemical reaction or the simulation of urban traffic are just some examples. We choose the definition established with general character in [30], and applied to the simulation of microworlds. We therefore define a cellular automaton (AC) as a 4-uple $M = (\Omega, \eta, \mathrm{N}^{\mathrm{I}}, \lambda)$, whith the following definitions:

- Ω is a grid of places (also called nodes or cells) together with some set of boundary conditions in the finite case, which define the neighbourhood of cells on the periphery of a give one; this grid forms the microworld.
- η is a an alphabet with a finite number of symbols (usually with conmutative ring structure) of states; cells can adopt values belonging to it.
- N^{I} is a finite set of cells that define the neighbourhood with a given cell interacts.
- λ is a transition function that specifies how a cell in the network changes its state as a function of time and neighborhood state N^{I}.

With the above definitions, a microworld can be defined as a grid Ω included in the real space Z^2 which homogeneously cover a portion of the Euclidean 2-dimensional space. Every cell is labeled by coordinates $s \in \Omega$. The spatial arrangement of cells is specified by connections to their closest neighbours, which are obtained by joining pairs of cells in some regular arrangement. For a spatial coordinates s, the neighbourhood network $N_b(s)$ is a list of neighboring cells defined by

$$N_b\,(s) = \{s + c_i{:}c_i{\in}N_b, i = 1, \cdots ,b\} \tag{1}$$

In equation (1), b is the coordination number or, to put it another way, the number of close neighbours in the grid that interact with the cell located in coordinates s. With N_b we denote the set of close neighbors with elements $c_i \in Z^2$, for $i = 1, \cdots, b$. In the case at hand, and for $d = 2$, the only regular polygons forming a regular tessellation of the plane are triangles ($b = 3$), rectángules ($b = 4$) and hexagons ($b = 6$), and we will choose for our model the second case, so that

$$\Omega = \left\{\mathrm{s} : \mathrm{s} = (\mathrm{s}_1, \mathrm{s}_2) \in \mathbb{Z}^2\right\} \tag{2}$$

Total number of available cells is usually anotated with $|\Omega|$. In computer simulations, ACs use grids that are necessarily finite ($|\Omega| < \infty$), and border conditions should be imposed establishing which are the neighbours of those cells located at the declared borders. In our case, we will use two types of boundary condition: null or cylindrical, according to the microworld that is pretended to simulate. The set of neighboring cells whose state influences in a given one, which is defined by the neighborhood of interaction $N_b^I(s)$ for a cell s, according to the following expression:

$$\mathrm{N}_{\mathrm{b}}^{\mathrm{I}}\,(\mathrm{s}) = \{\mathrm{s}{+}\mathrm{c}_{\mathrm{i}}{:}\mathrm{c}_{\mathrm{i}} \in \mathrm{N}_{\mathrm{b}}^{\mathrm{I}}\} \tag{3}$$

On the other hand, each cell $p \in \Omega$ has an assigned state $s(p) \in \eta$. Elements belonging to η can be numbers,

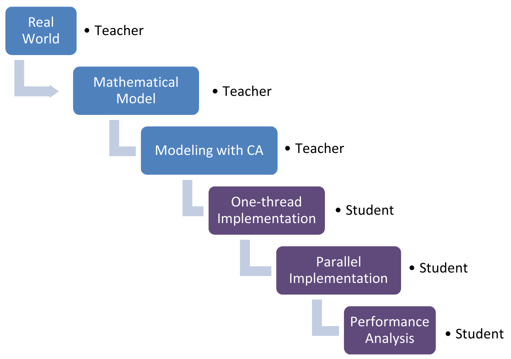


Fig. 1. Work flow for each microworld, with roles developed by the teacher and by the students. The instructor presents a real, mathematical and CA model, and students program it; first, tiwh a single thread approach, and second, with a multithread approach. Finally, run time and speedup are calculated.

letters or symbols. We will choose η depending on the environment to be modeled with the microworld. A configuration for the AC $c \in \eta^{|\Omega|}$ is determined by the state of all cells in the grid; model offers a snapshot of the state of the tumor at a given instant of time, and dynamically changes that state using a discrete time sequence according to the rule that the transition function imposes.

The temporal evolution of the grid is fixed by the transition function λ, which tells how a cell changes to a new state as a function of its previous state, and of its interaction with its cell neighbourhood, according to equation 4, where $\mu = \left|N_b^I\right|$.

$$\lambda : \eta^{\mu} \rightarrow \eta \tag{4}$$

The rule is spatially homogeneous, and therefore does not explicitly depend on the position r for a given cell

IV. Case Studies

We dedicate this section to describe the set of parallel programming projects based on microworlds that we have designed for our students as part of the practices of the academic year in which we have developed the experience. The description should be sufficiently detailed so that the experience can be reproduced from the text for interesting readers. In all cases, the scheme of the work followed by the instructors and students is illustrated in Fig. 1.

The teacher performs a previous task where he exposes the entity to be modeled to the students, and then extracts a continuous mathematical model, and ends up proposing the students a discrete mathematical model using a two-dimensional cellular automaton.

The students, using the Java programming language, implement the cellular automaton by means of a sequential code, and then obtain a parallel version, applying the data division scheme described in Fig. 2, together with different additional elements of parallelism control shown in Table I.

This table shows in column, for each one of the microworlds, the set of parallel programming topics that will be used in its implementation. So, for example, to implement *Life*, students use parallel tasks through the `Thread` class, but

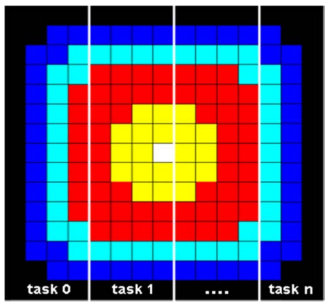


Fig. 2. Scheme for split of the data space of the microworld by a number n of parallel tasks. The students performed the division manually and automatically.

TABLE I
LEARNING TARGETS FOR MICROWORLDS

Target	Life	Wa-Tor	Tumor	Caves
Thread class	Yes	No	No	No
Runnable	No	Yes	Yes	Yes
Random class	Yes	Yes	Yes	Yes
Data partition[1]	Yes	No	No	No
Data Partition[2]	No	Yes	Yes	Yes
Mutex	No	No	Yes	No
Barrier	Yes	Yes	Yes	Yes
Thread Pool	No	Yes	Yes	Yes
GUI	Yes	Yes	Yes	Yes
Time measurements	No	No	Yes	Yes
Speedup	No	No	Yes	Yes

not through the `Runnable` interface, they generate random data using the `Random` class, they make a manual and non-automated division of the data space, they do not use mutual exclusion with locks to control data structures, nor a thread pool executor to process the tasks, althoug and synchronize them through the Barrier class; finally they implement a graphical interface, and they do not take execution times nor calculate speedup.

The simulations of the microworlds are not interactive, except for the simulation of the *Wa-Tor* planet, where the student can modify the parameters of the simulation during the course of the simulation. Each one of the microworlds proposed in the experience increases the difficulty of implementation with respect to the previous one, adding new complexities, using new classes and elements of the API to control concurrency and, in general, gradually increasing the learning of more complex concepts, and reinforcing by repetition those that are considered more important. The concepts of multicore parallel programming that we have tried to teach the students have been: techniques of creation of parallel tasks through inheritance of the Thread class and implementation of the Runnable interface, partition of the data domain (Fig. 2) manual and automatic, control of mutual exclusion on the data when it is necessary, use of executors, use of barriers, measures of run time, calculation of the speedup and representation by means of a graphical interface of the microworld.

[1]Manual division of the data domain via class constructor.

[2]Automatic division of the data domain with the method `availableProcessors()`.

TABLE II
PARAMATERS FOR LOTKA-VOLTERRA'S EQUATIONS

Parameters	Meaning
α	Fishes growth rate
β	Success of sharks hunting
γ	Rate of decline of sharks
δ	Food obtained by sharks

A. The Game of Life

Proposed in [5] by mathematician J.H. Conway, there is a microworld that simulates the interaction between a population of cells that can be in states $a_{i,j} \in \{0,1\}$. The transition function that students must apply to each cell of the micro-world can be described by equation (5). The students were asked to perform a parallel simulation of Life, fulfilling the objectives set out in Table I. In this phase of the experience, the students developed parallel code with manual partitioning of the data domain. Special emphasis was also placed on the design of a user interface that would allow a graphical visualization of the simulation, as shown in Fig. 3, where each generation shown should have been calculated using parallel threads.

$$\lambda\left(a_{i,j}^{(t+1)}\right) = \begin{cases} 1, & if\ a_{i,j}^{(t)} = 0 y \sum_{i=-1,j=-1}^{1,1} a_{i,j}^{(t)} = 3 \\ 1, & if\ a_{i,j}^{(t)} = 1 y \sum_{i=-1,j=-1}^{1,1} a_{i,j}^{(t)} < 3 \\ 0, & if a_{i,j}^{(t)} = 1 y \sum_{i=-1,j=-1}^{1,1} a_{i,j}^{(t)} > 3 \end{cases} \tag{5}$$

B. Sharks and Fishes in Planet Wa-Tor

Proposed in [8] by A.K. Dewdeney, Wa-Tor is a reticulated aquatic world of toroidal structure, inhabited by two species: sharks and fish. Each point of the Wa-Tor grid $a_{i,j} \in \{0,1,-1\}$. Fishes (-1) and sharks (1) follow the classical predator-prey dynamics described in reference [8] by the classical Lotka-Volterra's equations (equations 6 and 7).

$$\frac{dx}{dt} = \alpha x - \beta xy \tag{6}$$
$$\frac{dx}{dt} = -\gamma y + \delta yx \tag{7}$$

Here, x is the number of fishes, and is the number of sharks, and the coefficients have the meaning shown in Table II. Once interpreted discreetly [21], they describe a micro-world in which initially 50% of the grid points contain fishes, (-1) 25% contain sharks (1) and the rest are empty (0). Students must implement a simulation of the Wa-Tor planet (see Fig. 4), where the learning objectives are those described in Table I.

The students implemented the parallel simulation adding several improvements with respect to the previous micro-world, the most remarkable being the processing of the parallel tasks by delegating their life cycle to a thread pool executor.

In this way, they focused more on the coding of the structure and meaning of parallel tasks, and less on the management of their life cycle, which was developed automatically by the thread pool.

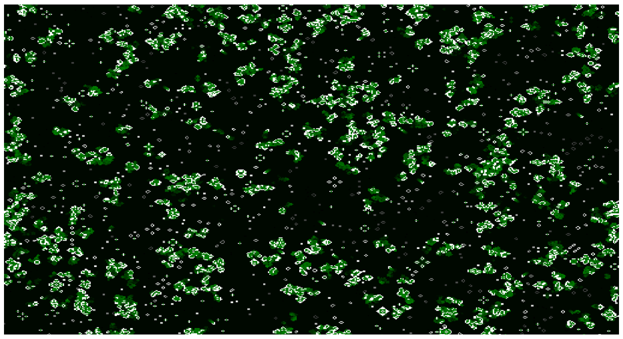
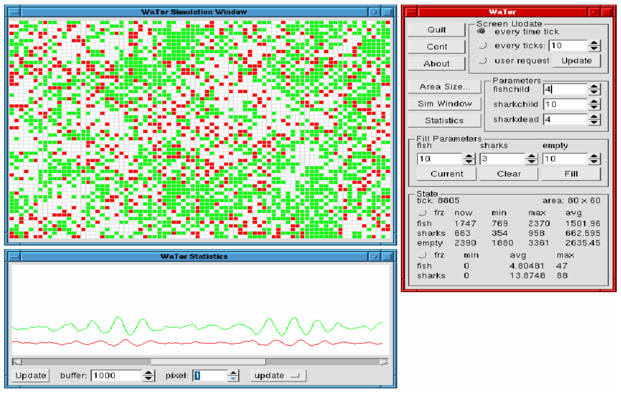


Fig. 3. Life. The state of the micro-world is illustrated after several generations of evolution from an initial random configuration. The formation of colonies of living cells is observed.



```
Screen Update
 (●) every time tick
 ( ) every ticks: 10
 ( ) user request   Update
Parameters
 fishchild  4
 sharkchild 10
 sharkdead  4
Fill Parameters
 fish   sharks   empty
 10     3        10
State
tick: 8805                     area: 80 x 60
 ( ) frz  now    min    max    avg
 fish     1747   769    2370   1501.96
 sharks   663    354    958    662.595
 empty    2390   1880   3361   2635.45
 ( ) frz  min    avg       max
 fish     0      4.80481   47
 sharks   0      13.8748   88

Update  buffer: 1000  pixel: 1  update
```

Fig. 4. The Wa-Tor micro-world. It illustrates the state of the micro-world (top left grid), the population curves that follow the dynamics described by the Lotka-Volterra's equations, and the window (located on the right side of the image) that allows you to adjust the simulation parameters in orde to have interactivity.

C. Tumor Growth Simulations

A tumor cell in this micro-world is an individual entity that occupies a point in a finite two-dimensional grid Ω, and can (see [30]) perform several different actions: move to another positions in the grid, replicate doing mitosis, die, or hold stable in the phase. G_0 of the cell cycle, where the cells are quiescent. Tumour cells are usually of the stem type and have unlimited proliferation capacity, and non-stem type (they can carry out a mitosis in the M phase (mitosis) of the cell cycle a limited number of times).

We will not assume stem cells, but will instead model whether or not a cell s lives by the probability distribution of equation 8.

$$\lambda_l\left(\mathrm{s_{N(r)}}\right) = \begin{cases} 1 & with\ probability\ \mathrm{W}\left(\mathrm{s_{N(r)}} \rightarrow 1\right) \\ \\ 0 & with\ probability\ \mathrm{W}\left(\mathrm{s_{N(r)}} \rightarrow 0\right) \end{cases} \tag{8}$$

The students had to model this distribution and the one described by equation 9 using as a random number generation tool the Java Random class, which allows, through multiple instances, to generate them in parallel.

The adjustment of this distribution allows us to have, if desired, also stem cells, and what is more important, it allows to model the survival of the cells caused by intrinsic

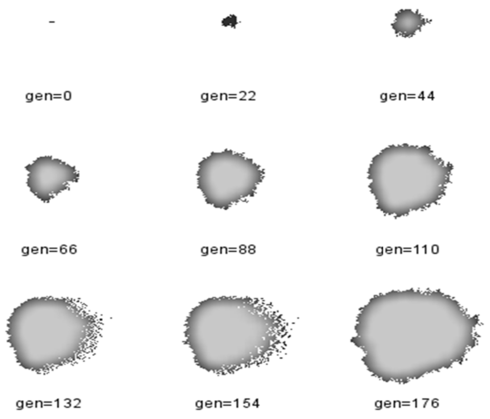


Fig. 5. This figure shows the evolution for the tumor growth simulation model withs a tissue of 128 × 128 cells and 200 generations of run time. Different snapshots of the state of the tumour are shown from an initial seed of four cells in different instants of time, which are indicated through the number of generation in which each of the snapshots was taken.

factors to the tumor (angiogenesis, tumor microenvironment, etc.) or to extrinsic factors of therapeutic type (response to cytostatics or cytotoxics, use of antiangiogenic drugs, radiation, interferons, etc.) adjusting the distribution according to the known tumor response either in vivo or in vitro, and contrasted in clinical or laboratory practice.

A living cell can replicate by generating a daughter cell by mitosis if there are free spaces available in its neighbourhood, a phenomenon that the model contemplates by means of a second probability distribution, conditioned to the previous one, as described in equation 9.

$$\lambda_p\left(\mathrm{s}_{\mathrm{N(v)}}\right)=\begin{cases}1, & \textit{with probability } \mathrm{W}'\left(\mathrm{s}_{\mathrm{N(v)}}\rightarrow 1\right)\\ 0, & \textit{with probability } \mathrm{W}'\left(\mathrm{s}_{\mathrm{N(v)}}\rightarrow 0\right)\end{cases} \tag{9}$$

Now, the position of the network $v \in s_{N(r)}$ will host the cell resulting from the mitosis of the r-cell, if there is space to host it.

$$\lambda_m\left(\mathrm{s}_{\mathrm{N(v)}}\right)=\begin{cases}1, & \textit{with probability } \mathrm{W}''\left(\mathrm{s}_{\mathrm{N(v)}}\rightarrow 1\right)\\ 0, & \textit{with probability } \mathrm{W}''\left(\mathrm{s}_{\mathrm{N(v)}}\rightarrow 0\right)\end{cases} \tag{10}$$

Lastly, a living cell can migrate, changing position within the tumour, provided there is space available for this. This has been contemplated by means of a third probability distribution, also conditioned to λ_l.

The model needs parameters for the probability distributions described in the previous section, which can be chosen by the user and allow to Monte-Carlo stochastic simulation to decide if a cell dies, remains in the phase G_0 of the cycle, or enters the M phase of the cycle and reproduces by mitosis, provided there is enough space around it to do so. The direction of propagation of the mitosis is decided by a probability distribution that chooses one of four possible directions.

On this occasion, the students received a pseudocode implementation that simulated tumor growth (see Fig. 5) by means of a sequential code [4] and a conditional type pattern (pre, post) as described in reference [4] and they had to implement a parallel simulation, where the learning objectives were those indicated in Table I for this micro-world.

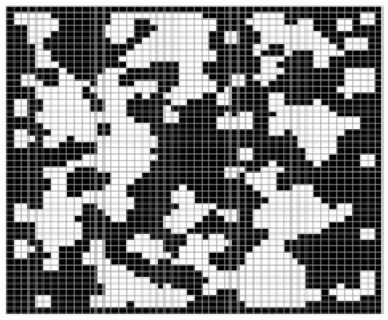


Fig. 6. Cave microworld in the form of a labyrinth resulting from applying the parallel simulation of the cellular automaton described in the text.

D. Generation of Caves (Cave)

In many videogames, it is common to use an environment with a labyrinth structure (Fig. 6), through which different actors move (the paradigm of this type of videogames is the venerable Pac Man). There are proposals in the literature [16] that use two-dimensional cellular automatons to generate a labyrinth that changes with each new game. The grid that models this microworld has cells that take values $a_{i,j} \in \{floor, wall, rock\}$. Initially the grid is filled with cells with states $a_{i,j} = floor$, and then a generator of uniformly distributed random numbers is used to convert the state of the $r\%$ from cells to $rock$. Subsequently, the grid is processed in parallel. In each generation, the transition function of equation (11) is applied to all cells of the grid. The T and r parameters are set by the user according to the final complexity desired for the labyrinth; this model has been applied in the field of video games, and T and r depends on the level at which the player is. In general, for a higher level, complexity is greater.

$$\lambda\left(a_{i,j}^{(t+1)}\right)\begin{cases} rock & \textit{if } a_{i,j}^{(t)}=\textit{floor and } \sum_{i,j=-1}^{1} a_{i,j}^{(t)} \geq T\\ a_{i,j}^{(t)} & \textit{if } a_{i,j}^{(t)}=\textit{floor and } \sum_{i,j=-1}^{1} a_{i,j}^{(t)} < T\end{cases} \tag{11}$$

$$\lambda\left(a_{i,j}^{(t+1)}\right)=\begin{cases} wall & \textit{if } a_{i,j}^{(t)}=\textit{rock and } N_f \geq 1\\ a_{i,j}^{(t)} & \textit{if } N_f < 1\end{cases} \tag{12}$$

Once the labyrinth has been built after n iterations, it is time to build the walls, for which the transition function of equation 12 is applied only on those cells that are of class $rock$. In equation 12, N_f is the number of neighboring cells that are of type $floor$.

Of course, the method proposed here would admit -and indeed require- adding the processing necessary to ensure that the labyrinth generated is viable. That is, it would be necessary to add a routine that looks for valid paths in the generated labyrinth, and discards it if there are none. Such a procedure is perfectly characterized in the literature [12] and is easy to implement, although it was not posed to our students.

TABLE III
CONTENTS FOR THEORETICAL COURSE (FROM ACM/IEEE GUIDES)

Topic	Time (hours)
Multiple simultaneous computations	8
Parallelism versus concurrency	4
Communication and coordination	10
Race Conditions	4
Vivacity and progressivity	4

V. THE EXPERIMENT

In order to develop the experiment, the course “Introduction to Parallel Programming” was chosen as the context. It was followed by the students of the degree in Computer Scence of the University of Cadiz (Spain), in the fifth semester of the degree, with an academic load of 6 ECTS credits, 3 theoretical credits and 3 practical credits.

The students accessed the course having previously followed (third semester) a “Concurrent and Real Time Programming” course, and had elementary knowledge about the creation, management and control of concurrent threads with the Java language. However, they had no knowledge of multicore parallel programming.

They also had no previous experience with learning programming with microworlds. The 30 hours of practice were divided into four blocks of 7.5 hours, during each of which the students developed a mini programming project, with each of the microworlds described, following the work diagram proposed in Figure 1.

They also had to dedicate personal work time at home to each of them. The 30 hours of theory were configured according to the recommendations of the ACM/IEEE curriculum guide [1], as described for the “PD/Parallelism Fundamental” course as illustrated in Table III.

Fifty-two students (42 men and 10 women) with an average age of 21.84±0.63 years were enrolled in the course, who followed the internships through the microprojects of programming described in section IV. Four students dropped out of the group and failed (including dropouts) eight students.

As a control group, enrolled students (46 students, 37 men and 9 women), with an average age of 22.35±0.46 years, were used for the previous academic year, who followed the practices using the standard model and implemented classical parallel algorithms (parallel product of matrix, worker-slave model, etc.); 8 students dropped out. At the beginning of the course our students filled a survey where some demographic variables were measured, such as age and sex. At the end of the semester, the practical grades of the students of the experimental group were obtained. Main descriptors of the experience, once the dropouts were excluded, are shown in Table IV. These items were available for the students of the previous academic year, who acted as a control group. In addition, using the digital platform Moodle as support, the students completed a questionnaire whose variables are listed in Table V. In order to measure the benefits of the new parallelism teaching model and the degree of achievement of the objectives A, B and C of the experience, three prospective frameworks have been used, which are described below:

TABLE IV
MAIN VARIABLES FOR THE EXPERIMENT

Variable	Experimental	Control
N (Excluded dropouts)	48	38
Sex (M-F)	39-9	30-8
Dropout (%)	7.69%	13.6%
Average Rating	7.03±1.88	6.31±1.62
Fail (%)	18.18%	21.05%

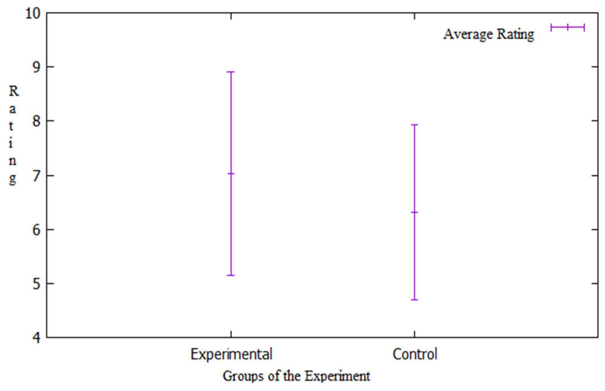


Fig. 7. Average rating along with the standard deviation in the experimental and control groups.

- Measure of the average rating obtained by the group of students in the practices of the course developed with microworlds, represented by a numerical value in the interval [0-10], compared with the same value obtained by the students in our control group (Objective A).
- Measurement of the degree of effort of the students with the new proposed model of practices, versus the standard model, measured in hours of dedication to the practices of the course (in classroom and at home), (Objective B).
- Measurement of students’ subjective impression of their personal perception of parallel programming learning, measured through an instrument in the form of a multiple item survey, and assessment in the range [0-6], (Objective C).

VI. ANALYSIS AND DISCUSSION

The results obtained in the first of the measurement scenary (Fig. 7) showed how students who followed the practice model based on microworlds obtained an average practice score of 7.02±1.88 points, compared to students in the control group, who obtained an average score of 6.31±1.62, fails is a score lower than 5.0 points in both cases.

The % of students who did fail the practices of the course was 18.8% in the experimental group, compared to 21.05% in the control group. We used the Shapiro-Wilk test applied to the data samples of both groups for the average rating variable. Neither the experimental group ($W = 0.8545$, $p < 0.001$) nor the control group ($W = 0.8880$, $p < 0.001$) were normal for this variable, with $\alpha = 0.05$.

Attributing to chance the difference in ratings between both groups of students (H_0), Mann-Withney’s statistic U ($U = 434.500$, $p = 0.033$) showed that this difference is not due to

TABLE V
SUBJECTIVE VARIABLES MEASURED

Variable	Meaning
q1	I know to create threads by inheritance of the Thread class
q2	I know to create tasks implementing Runnable interface
q3	I know how to split the data domain manually
q4	I know how to split the data domain automatically
q5	I can synchronize with a cyclic barrier protocol.
q6	I know how to control access to data with mutual exclusion
q7	I know how to use a thread pool executor
q8	The micro-worlds stimulate me to do the practices
q9	I prefer to program micro-worlds in the practices

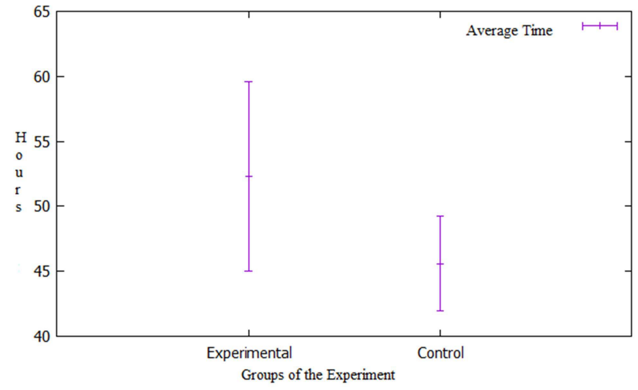


Fig. 8. Average time and standard deviation of working hours of practices in experimental and control groups.

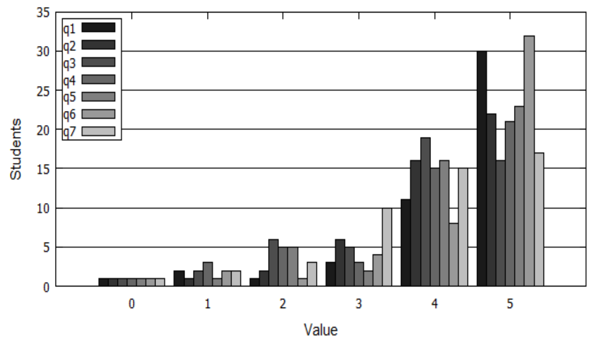


Fig. 9. Results of the evaluation of the model based on the subjective variables q1 to q7 listed in Table V.

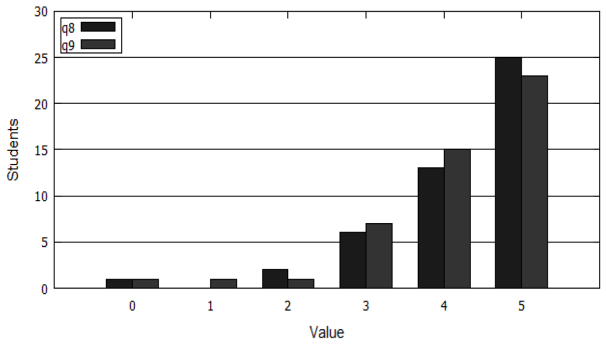


Fig. 10. Results of the evaluation of the model based on the subjective variables q8 and q9 listed in the Table V.

chance, and is attributable to the proposed method of learning parallelism. A statistically significant improvement for average results of the students in the development of the practices of the subject and that shows that the experience fulfilled the objective A is observed. For the second scenario and objective B, students were asked to keep a log of the total time devoted to the development of the subject practices, in the academic centre and at home.

The average results obtained are illustrated in Fig. 8. It can be seen that the proposed model needs to be developed a total number of hours (52.3±7.32) of dedication discreetly greater than the time required when using the standard model (45.6±3.66). The Shapiro-Wilk normality test was applied to the working hours of the students in the experimental and control groups.

Neither the experimental group ($W = 0.7963$, $p < 0.001$) nor the control group ($W = 0.8125$, $p < 0.001$) were normal for this variable, with $\alpha = 0.05$. Attributing to chance the difference in student working hours (H_0), Mann-Withney's statistic U ($U = 346.272$, $p = 0.029$) showed that this difference is not due to chance, and that the additional effort required is attributable to the proposed method of learning parallelism.

For the third scenario, a survey ($n = 48$) of multiple variables was used (Table V) to measure the subjective impression of the students and check the C objective, was carried out by them at the end of the semester anonymously, measuring, on a discrete scale of values between 1 (not at all in agreement) and 5 (very much in agreement); the value of 0 was used to represent do not know/not answer. The results of the survey are represented graphically in Figure 9, which shows, for each of the variables that measure learning aspects related to parallel programming (q1 to q7), how students scored it in each of the six possible scoring categories (range 0-5). It can be seen that in practically all of the variables used, the students scored them mainly with values located in the upper part of the rating scale (values 4 and 5), which provides evidence that the subjective impression about the learning of main topics of parallel programming is reasonably good.

In addition, Fig. 10 shows the subjective scores of the variables q8 and q9, which measure whether students have felt stimulated to commit themselves to do the practices of the course, in the sense proposed in [18], and their overall satisfaction with the experience. In both cases the results offer high scores indicating that the stimulus has existed (q8), and that their overall satisfaction (q9) with the experience is high.

VII. CONCLUSIONS AND FUTURE WORK

The analysis of the measures carried out in the previous section offers a sufficiently significant statistical validity to consider the proposed model viable, depending on the objectives envisaged. It should be remembered that these were materialized in order to make students to learn a minimum body of knowledge that was sufficiently broad and deep to face the development of parallel solutions in the field of multicore platforms, using the Java programming language, with a reasonable degree of effort, and increasing the degree of commitment of the student to his or her learning. The discussion of the results has shown that all of this is possible. Specifically, it has been possible to measure a positive impact of the experience in the average ratings that students obtain in the practices of the course.

The subjective survey that has measured the improvement of learning for different aspects of multicore parallel programming (Fig. 9) confirms that students' scores are "high" or "very high" mostly. Another of the achievements of the experience has been a greater stimulus among students at the time of committing themselves to the development of the practices of the subject, which is specified as we have already in a higher mean score (7.02±1.88 vs. 6.31±1.62) for students who programmed microworlds (experimental group) versus those who did not (control group), with a discrete improvement in the rate of students who score with fail.

The reinforcement that the students received to commit to the development of the practices, measured with item q8 (Fig.10) shows again high scores on that commitment.

We believe that this overall improvement in the results obtained is mainly due to the novel factor that ludification introduces in itself, and to the attractiveness that being able to implement microworlds and see them in action, offers students. In both ways, they obtain a stimulus in the form of a visual reward that they may not have, or have to a lesser extent with other practice approaches; this stimulus acts as a positive reinforcement in the student's commitment to their learning, and improves their general attitude towards the work they have to do. The main negative aspect that we identify is the greater dedication to the practices of the subject that the students need (Fig. 8), although it should be noted that this has always been within the number of ECTS credits that the subject has been assigned. As future line of investigation we are considering the extension of the model to microworlds supported by manycore architectures of GPU typology and the application of the model in the development of the practices of the course "Concurrent and Real Time Programming" (third semester for the Computer Science degree in the University of Cádiz).

References

[1] M. Sahami *et al.*, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: ACM, 2013.

[2] E. Buzek, "Web platforms for parallel programming tutorials," Digitalní Repositár UK, 2017. [Online]. Available: https://www.natur.cuni.cz/geografie/mapova-sbirka/digitalni-univerzitni-repozitar

[3] Y. Chou, *Actionable Gamification: Beyond Points Badges and Leaderboards*. Fremont, CA, USA: Octalysis Media, 2016.

[4] M. I. Capel, A. J. Tomeu, and A. G. Salguero, "Teaching concurrent and parallel programming by patterns: An interactive ICT approach," *J. Parallel Distrib. Comput.*, vol. 105, Jul. 2017, pp. 42–52 2017, doi: 10.1016/j.jpdc.2017.01.010.

[5] J. H. Conway, "Mathematical games: The fantastic combination of John Conway's new solitaire game life," *Sci. Amer.*, vol. 223, pp. 120–123, Oct. 1970.

[6] E. L. Deci, R. Koestner, and R. M. Ryan, "Extrinsic rewards and intrinsic motivation in education: Reconsidered once again," *Rev. Educ. Res.*, vol. 71, no. 1, pp. 1–27, Mar. 2001.

[7] P. Denny, "The effect of virtual achievements on student engagement," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst*, 2013, pp. 763–772.

[8] A. K. Dewdney, "Computers recreations sharks and fish wage an ecological war on the Toroidal planet Wa-Tor," *Sci. Amer.*, vol. 251, pp. 14–22, Dec. 1984.

[9] P. Díaz, *Faltan 900.000 Profesionales TIC*. Diario El Mundo, edición digital, 2015.

[10] P. Fotaris, T. Mastoras, R. Leinfellner, and Y. Rosunally. (2015). *Who Wants to be a Phytonista: Using Gamification to Teach Computer Programming*. [Online]. Available: http://www.academia.edu/download/44361560/Who_wants_to_be_a_Pythonista_Using_Gamif.pdf

[11] F. Gallego, C. Villagrá, F. Llorens, and R. Carmona, "PLMan: A game-based learning activity for teaching logic thinking and programming," *Int. J. Eng. Edu.*, vol. 33, no. 2B, pp. 804–815, 2017.

[12] S. Gunuc and A. Kuzu, "Student engagement scale: Development, reliability and validity," *Assessment Eval. Higher Edu.*, vol. 40, no. 4, pp. 587–610, May 2015.

[13] J. Hamari, J. Koivisto, and H. Sarsa, "Does gamification work?—A literature review of empirical studies of gamification," in *Proc. 47th Hawai Int. Conf. Syst. Sci.*, Jan. 2011, pp. 1–10.

[14] W. Hsin *et al.*, "A practitioners guide to gamification of education," Behavioural Econ. Action. Rothman School Manage. Toronto, Toronto, ON, Canada, Tech. Rep., 2013. [Online]. Available: https://inside.rotman.utoronto.ca/behaviouraleconomicsinaction/files/2013/09/GuideGamificationEducationDec2013.pdf

[15] M.-B. Ibáñez, Á. Di-Serio and C. Delgado-Kloos, "Gamification for engaging computer science students in learning activities: A case study," *IEEE Trans. Learn. Technol.*, vol. 7, no. 3, p. 291-pp 291-301, Jul. 2014.

[16] L. Johnson, G. Y. Yannkakis, and J. Togelius, "Cellular automata for real-time generation of infinite cave levels," in *Proc. Workshop Procedural/Content Gener. Games*, Jun. 2010, p. 10, doi: 10.1145/1814256.1814266.

[17] A. Knutas, J. Ikonen, U. Nikula, and J. Porras, "Increasing collaborative communications in a programming course with gamification: A case study," in *Proc. 15th Int. Conf. Comput. Syst. Technol. CompSysTech*, 2014, pp. 370–377.

[18] M. Mallon, "Gaming and gamification," *Public Services Quart.*, vol. 9, no. 3, pp. 210–221, Jul. 2013, doi: 10.1080/15228959.2013.815502.

[19] P. E. McKenney, "Is parallel programming hard, and, if so, what can you do about it (v2017.01.02a)," 2017, *arXiv:1701.00854*. [Online]. Available: http://arxiv.org/abs/1701.00854

[20] P. Mozelius *et al.*, "Game-based technologies in teaching programming in higher education: Theory and practices," *Recent Patents Comput. Sci.*, vol. 9, no. 2, pp. 105–113, Jul. 2016.

[21] S. Olek, "An accurate solution to the multispecies Lotka–Volterra equations," *SIAM Rev.*, vol. 36, no. 3, pp. 480–488, Sep. 1994.

[22] N. Paspallis, *A Gamification Platform for Inspiring Young Students to Take an Interest in Coding* (Information Systems Development: Transforming Organisations and Society through Information Systems), V. Strahonja, N. Vrček, D. Plantak Vukovac, C. Barry, M. Lang, H. Linger, and C. Schneider, (Eds.), 2014.

[23] M. Ortiz-Rojas, K. Chiluiza and M. Valcke, "Gamification in computer programming: Effects on learning, engagement, self-efficacy and intrinsic motivation," in *Proc. Eur. Conf. Games Based Learn.*, 2017, pp. 507–514.

[24] M. Ortiz, K. Chiluiza, and M. Valcke, "Gamification in higher education and STEM: A systematic review of literature," in *Proc. 8th Int. Conf. Educ. New Learn. Technol.*, 2016, pp. 6548–6558.

[25] C. R. Prause and M. Jarke, "Gamification for enforcing coding conventions," in *Proc. 10th Joint Meeting Found. Softw. Eng. ESEC/FSE*, 2015, pp. 649–660, doi: 10.1145/2786805.2786806.

[26] J. Rojas and G. Fraser, "Teaching mutation test using gamification," in *Proc. Eur. Conf. Softw.*, 2016, pp. 401–408.

[27] N. Sahari, T. S. Meriam, T. Wook, and A. Ismail, "The study of gamification application architecture for programming language course," in *Proc. 9th Int. Conf. Ubiquitous Infomation Managment Commun.*, vol. 17, pp. 1–5, Jan. 2015.

[28] M. Sanmugam, Z. Abdullah, and N. M. Zaid, "Gamification: Cognitive impact and creating a meaningful experience in learning," in *Proc. IEEE 6th Conf. Eng. Edu. (ICEED)*, Dec. 2014, pp. 123–128

[29] J. Simoes, R. D. Redondo, and A. F. Vilas, "A social gamification framework for a K-6 learning platform," *Comput. Hum. Behav.*, vol. 29, no. 2, pp. 345–353, Mar. 2013.

[30] A. J. Tomeu, A. G. Salguero, and M. I. Capel, "Speeding up tumor growth simulations using parallel programming and cellular automata," *IEEE Latin Amer. Trans.*, vol. 14, no. 11, pp. 4611–4619, Nov. 2016, doi: 10.1109/TLA.2016.7795837.

[31] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in *Proc. Conf. Innov. Technol. Comput. Sci. Edu. (ITiCSE)*, 2014, pp. 39–44.

[32] G. Zichermann and C. Cunningham,*Gamification Design: Implementating Mechanics in Web Mobile Apps*. Sebastopol, CA, USA. O'Reilly, 2011.

Antonio J. Tomeu Hardasmal received the B.Eng. and M.Eng. degrees in computer science from the University of Granada, Spain, in 1990 and 1992, respectively, and the Ph.D. degree in mathematics from the University of Cádiz, Spain, in 2002. He is currently an Associate Professor of computer science with the University of Cádiz. His current research interests are applications of parallel programming techniques to simulate natural phenomena and physical modeling with cellular automaton. He is a Coordinator of Red Iberoamericana de Investigación en Tecnologías Concurrentes, Distribuidas y Paralelas (TECDIS) and the Editor-in-Chief of *Annals of Multicore and GPU Programming*.

Alberto G. Salguero received the B.Eng. and M.Eng. degrees in computer science from the University of Granada, Spain, in 2002 and 2004, respectively, and the Ph.D. degree in computer science from the University of Cádiz, Spain, in 2013. He is currently an Associate Professor of computer science with the University of Cádiz. His current research interests are applications of parallel programming techniques to simulate natural phenomena and ontologies applied to data integration. He is a member of Red Iberoamericana de Investigación en Tecnologías Concurrentes, Distribuidas y Paralelas (TECDIS).