Proceedings of the Fourteenth Artificial Intelligence and
Interactive Digital Entertainment Conference (AIIDE 2018)

Evolving Behaviors for an Interactive Cube-Based Artifact

Victor M. Oliveira,[1] **Hugo A. D. do Nascimento,**[1]
Fabrizzio A. A. M. N. Soares,[1] **Cleomar S. Rocha**[2]
[1]Instituto de Informática, Universidade Federal de Goiás, Goiânia, Goiás - Brazil
[2]Faculdade de Artes Visuais & Media Lab, Universidade Federal de Goiás, Goiânia, Goiás - Brazil
{vtrmartin,cleomarrocha}@gmail.com, {hadn,fabrizzio}@inf.ufg.br

Abstract

In the present paper we explore the idea of combining computation power and the availability of ordinary art spectators in order to produce new interactive art works. This is investigated for a particular application, which consists of producing new behaviors for a programmable art apparatus named C^3 *Cubes*. Given the nature of the problem and some difficult challenges to be dealt with, an Interactive Evolutionary Computation (IEC) approach was devised. Furthermore, it was necessary to adopt a surrogate function method for approximating the user's preferences and to implement a Web-based virtual simulation environment for speeding up the generation and the evaluation of C^3 *Cubes* projects. The integration of all these elements is crucial for producing new user-guided cube projects with interesting behaviors. The main approaches experimented in this research and the proposed design solutions are useful to solving similar problems in other domain areas, for example, in the context of game design.

Introduction

The advances in technology in the last 40 years, such as the development of new types of sensors and visual displays as well as of cheaper and yet more powerful processing devices, has made possible the growth and the popularization of technological art. Nowadays, many art events, museums and even musical and dance shows implement projects of technological art, attracting and captivating its spectators. Furthermore, research in Computer Science and in Engineering involving fields like computer programming, HCI and computer networking, among other important ones, paved the way to interactive technological-based art works, for which humans are not mere spectators but an essential part (called here interactors) of the art project. Thus, different interactive art projects have been presented by many artists, demonstrating how creative and effective in producing human experiences this area can be. For instance, Better Hands (Lages, Gobira, and Marinho 2017) is a work in which the spectators use a device to collect data from their physical hands. The data is then "transported" to a robot arm with a brush, that uses it as signal to paint in a tablet. In Tangible Air (Röggla et al. 2017), *Galvanic Skin Response* sensors are used to assess a person's reaction when observing a live event. The data from the sensors are processed and showed in a live visualization that presents how the spectators were engaged in the event.

The design of an artistic work is, in general, fully made by its creators. However, in some cases, the artists present an unfinished work and leave to the spectators the task of composing the final art by manipulating and combining a set of predefined elements. The human task can be as distinct as finishing a painting or building a physical structure by putting objects together. If computational resources are employed in the art project, then a myriad of new interactive approaches and feedback are possible. We have been interested particularly in studying ways of joining computation resources and the availability of ordinary spectators in order to produce new artistic works. The motivation is that spectators, even though they may not know how to design creative and engaging works of art, are able to experiment an art project and give a valuable impression about it. Computers, on the other hand, can explorer and build many different designs for an artistic project, in a well-defined scope.

We have experimented this idea with the C^3 *Cubes*, a technological apparatus developed by Media Lab of Federal University of Goiás (UFG) for producing artistic and ludic projects for exhibitions, fairs and events of other natures. The apparatus consists of three medium size cubes (60 cm side) that can be programmed to react to interaction. The cubes communicate among themselves and can be manipulated by one or more people simultaneously. The spectators can hold, shake, rotate and move the cubes, which results in feedback in the form of light changes and/or sounds. Each C^3 cube has its own behavior, which is encoded as a state machine. New behaviors can be programmed by specifying new state machines. The first demonstration of the C^3 *Cubes* was at the "Understanding Visual Music" event held in Brasília, Brazil, in 2015. For that event, some words and symbols were painted on the external cover of the cubes. In addition, reactions (in the form of lights and sounds) were programmed to indirectly guide the interactors to place the cubes close together and with some specific sides facing up. Because of the large number of internal states and state transitions that can be combined, it is possible to program complex behaviors in the C^3 *Cubes* and thus meeting various needs, including the development of some games.

In their original proposal of usage, the behaviors for the C^3 *Cubes* and the painting of their external covers are often designed by artists. The behaviors are then programmed and uploaded to the devices by computer experts. In the present work, we depart from that approach: instead of manually designing and coding the state machines, we explorer the possibility of having a computer creating new, unseen and potentially interesting behaviors for the C^3 *Cubes*. Unfortunately, there are some challenges in following this approach:

- The definitions of "interesting" and "creative", when referring to the C^3 cube behaviors, are subjective, imprecise and hard, if not impossible, to formalize in a mathematical/computation way. This means that the human interactor is still important and should be closely involved in the process of producing new behaviors, for guiding it;
- Evaluating a physical C^3 *Cube* project may demand several minutes, since it is necessary firstly to upload the behaviors to each one of three cubes and then to play with the apparatus for some time in order to get an proper impression of it. If many projects have to be evaluated, then a strategy is needed in order to speed up this processing;

The solution for dealing with the first challenge is to employ Interactive Evolutionary Computation (IEC) for creating new behaviors for the C^3 cubes in a semiautomatic way. We adopt a specific approach in which the computer is responsible for generating and evolving a population of C^3 *Cubes* projects (each project consisting of three cubes), and the users (the interactors) are responsible for evaluating the projects (the individuals of the population), as commonly done in other IEC applications (Semet 2002). In addition, the results of the evaluating processes have to be quantified, although they may remain subjective.

The simple use of IEC, on the other hand, as it is the case for most population-based optimization method, implies in many C^3 *Cubes* projects to be evaluated. This causes a considerable fatigue if done by just one interactor. To address this issue, and also the second challenge listed above, two approaches were devised: (1) in some iterations of the IEC method, the user evaluation task is replaced by a surrogate function that tries to model the user's preferences [1]; (2) a virtual environment for simulating a C^3 *Cubes* project, integrated with the computational evolutionary process and available on the Web, is employed. By using this Web application, several users can simultaneously and more quickly experiment with the C^3 cubes and evaluates their behaviors.

The remainder of this paper explains the details of our approach. In the Background Section, we present a brief description of the C^3 Project and an overview of IEC; Next, we introduce our evolutionary approach and describe the virtual environment with the Web application developed for evolving C^3 projects; We then describe some experiments conducted with the evolutionary approach and discuss the obtained results; At last, we draw our main conclusions and point to new ideas for continuing improving this work.

[1] A surrogate is an approximation model that mimics a desired objective function, but with a lower computational cost (Kim et al. 2014). It is employed when the original objective function is too costly to be evaluated many times or when it is a black-box, that is, the original function is unknown.

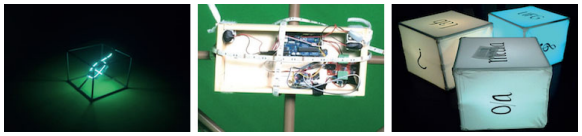

Figure 1: C^3 *Cubes* Project. From left to right: the physical structure of a cube without its Lycra cloth; the control box in the center of a cube; the complete set of cubes in exhibition.

Background

The C^3 Project

An illustration of the C^3 *Cubes* Project can be seen in Figure 1. The C^3 cubes have three main parts:

External/physical structure Each cube is an hexahedron with 60 cm of side made, currently, of PVC pipes. Two central PVC axis creates a place for laying a box with electronic devices. A white Lycra cloth is used for covering the cubes' surface. Symbols, words and drawings can be painted on the cloth, thus extending the meaning of the cube.

Electronic part A box in the inner part of the cube holds the electronic devices responsible for sensing the user interaction, generating audio and visual feedback, and for communicating with the other cubes. At the moment, the electronic components are an Arduino board, a accelerometer/gyroscope, a radio frequency emitter/receiver, a buzzer, a RGB LED tape, and other auxiliary components like batteries, cables and connectors.

Controlling software Each cube is controlled by a piece of software loaded into its Arduino. The software implements a *state machine* (composed of states and transitions) which is responsible for the cubes' behaviors. A *state* specifies none, one or more predetermined *actions*, such as switching on or off the light of the LED tape, changing its color or making a sound. Some actions have parameters like the RGB color for the LED lights and the frequency and the duration of a sound beep. All actions defined in a state must be executed when that state is reached. *Transitions* connect pairs of states in an orientated way and have one or more *conditions*. Examples of conditions are: if a certain amount of seconds has elapsed; if a given face (from 1 to 6) of the current cube is up; if the current cube is being held or shaken; if one of the other cubes is nearby; or if the current cube is in a certain state. The transitions that leave the current state are continuously verified in the software. If one of those transitions has all its conditions verified as true, then the machine swaps to the state indicated by that transition. We then say that the transition was *triggered*.

IEC

In Evolutionary Computing (EC), the fitness functions are often mathematically well-defined. Thus, for more subjective problems such as those in the arts and design fields,

EC becomes counterproductive. An alternative approach in this case is to adopt Interactive Evolutionary Computation (IEC), in which the human takes part in the evolutionary process, usually evaluating computer-generated solutions (Takagi 2001). Through IEC, it is possible to address subjectiveness and to incorporate some of the objectives and constraints that exists only in the mind of the spectator. Nevertheless, the use of human beings in an evolutionary processes also encounters challenges. In particular, the human fatigue makes it impossible for the population to evolve over many generations. Therefore, IEC is in general used with a small population and involves only a few generations (between ten and twenty generations). Certain mechanisms such as elitism and a proper choice of the evolutionary parameters help to accelerate the convergence of the evolutionary process, compensating for such disadvantages. As mentioned before, in the introduction section, we focus here on the use of a surrogate function for achieving this goal.

Despite the inherent limitations of IEC, it has been used successfully in several projects. For example, Hastings et al. (Hastings, Guha, and Stanley 2009) employed IEC in the evolution of elements of a computer game, with the aim of making it more interesting. Romero and Machado (Romero and Machado 2007) surveyed several cases of IEC for arts and aesthetics, concerned with the creation of images, music, and architectural designs, among other elements. Karl Sims (Sims 1991) and Draves (Draves 2005) applied IEC to generate images. Madera et al. (Madera et al. 2016) adopted IEC to evolve advertisement texts, whereas Makiwan et al. applied it for the selection of color pallets (Makiwan, Yoshida, and Koppen 2017).

The C^3 Evolutionary Approach

We now introduce our IEC approach for creating behaviors for the cubes of a C^3 Project. As described early, such a behavior is expressed as a state machine. From now on, we use the term "solution" (and "individual") to refer to a set of three state machines, one for each cube, that composes a complete C^3 *Cubes* project.

The IEC implements a standard evolutionary cycle with the following steps: (1) an initial population of individuals are created; (2) the individuals of the current population are evaluated; (3) a selection process of some promising individuals takes place; (4) crossover operators are applied on the selected individuals resulting in a set of new individuals; (5) the new individuals undergo mutation operations; and (6) the resultant individuals, possibly joined also with some unchanged individuals, form the new population that replaces the previous one. Steps (2) to (6) are repeated iteratively until a stop condition is reached. The difference here to a traditional evolutionary cycle is that step (2) above is mainly done by human interactors. As it will be seen later, this may also be performed automatically using a surrogate function.

Solution Representation

Individuals in a population are encoded using a high-level graph-based representation. An individual has three graph structures, each one defining a state machine (for a cube). The nodes and the edges of a graph represent the states and the transitions of the corresponding state machine, respectively. Each node has also a (possibly empty) list of feedback actions. Similarly, each edge has a list of conditions that define its transition.

By an automatic process, every correct solution representation can be translated into a state machine coded in a programming language. It can then be compiled and upload to the Arduino boards of the C^3 *Cubes*, for been experimented and evaluated.

Solution Evaluation

In the first tests of our approaches, we let the human interactors evaluate a C^3 *Cubes* project simply by giving a score to it between 1 and 10 (lower scores were considered better, as we aimed at a minimization problem). However, a plain numerical score is not a natural way of evaluating an art project. A numerical score is not even of much utility, since it does not provide details about the aspects that the person liked or not about the cubes.

We then changed the evaluation process to ask the interactors to express how they feel about three different aspects of the project: (A) the feedback (lights and sounds) produced the cubes; (B) the interaction among the cubes (for example, the actions performed by a cube when another cube was manipulated by an interactor); and (B) the general complexity of the behaviors implemented in the project. We also adopted a visual scheme consisting of thumbs up/middle/down for the interactors to provide their evaluation for each of the three aspects. Internally to the system, a thumb up, a thumb middle, and a thumb down are represented by the grades -1, 0 and 1, respectively. When summing up these grades for the three aspects, we get a value in the range from -3 to 3 that is then normalized, resulting in a positive score. Although we have ended up again with a numerical simple score, this new evaluation approach collects more information from the interactor about the C^3 *Cubes* project, which is used in a fine tuning control of the mutation operators, as explained later.

Another important element to mention is that, instead of showing just one solution for the interactor to evaluate, we always present a subset of the population (in the present work, we show five individuals at a time). This is because there is no precise reference measure of quality for the solutions. Therefore, it is difficult for an interactor to evaluate a solution by visualizing and testing only it. On the other hand, by having access to more than one solution, the interactor may compare them and define a scale of grades that are more intuitive for him/her. This also allows a better differentiation of the solutions.

Finally, in order to reduce the subjectiveness of the evaluation process, we demands every solution to be evaluated by more than one interactor (two interactors at the moment) and compute the average score of them.

Initial Population

The initial population is generated with diversified C^3 *Cubes* Projects. This is done by a combination of four small

manually-designed projects with some other randomly generated solutions.

Selection Process

After the individuals are evaluated, a quarter of the population with the highest scores are selected, and a roulette method is used to select another quarter, thus resulting in half of the population been selected. Then a pair for each individual, from the whole population, is chosen by a roulette method. The roulette method uses the score to decide the chance of a solution to be selected for forming a pair.

Crossover Operator

The crossover operator is applied to each pair defined by the selection process, resulting in two new solutions. For the present research, a two random point crossover operator was designed. Let (A, B) be a pair of individuals produced by the previous step. Each cube from A pairs with a random cube from B. Suppose A_1 and B_2 are the chosen cubes. Now, all states of A_1 are listed. This list is sliced in two random points, generating three sets of states. The same process is done for B_2. The sets of states from these different cubes are then aggregated, generating the new cubes D_1 and E_2. The cube D_1, for example, has some parts of A_1 and some parts of B_2, while E_2 has the remaining parts. This process is repeated for the other cubes of A. During the crossover, some transitions may loose their final state (they may point to a moved state). If this happens, then the final states of such transitions need to be redefined in order to point to an existing state (any one) in the new list.

Mutation Operators

After the crossover step, the resulting individuals undergo a mutation phase. There are 12 possibles mutations: creating a new state, deleting a state, adding a new action to a state, deleting an action of a state, changing the parameter value of an active action in a state (e.g., the RGB parameter of an action that sets the LEDs' colors), adding a transition, deleting a transition, changing the final state of a transition, changing the initial state of a transition, adding a new condition to a transition, removing a condition of a transition, and changing the parameter value of a condition in a transition. Transitions that loose their final state during mutation are also corrected, as done in the previous step.

Each mutation operator has a random probability of being applied to an individual, that is previously defined and very low (usually a value like 0.01). Nevertheless, those probabilities may change based on the average scores received by the population of individuals in the current evolutionary cycle. If the average score of the feedback aspect of the whole population is low, then the probabilities of the mutation operators for adding actions and changing their parameters are momentarily increased. If the average score of the interaction aspect is low, then the activation probabilities of the mutations that add a transition and change their final states are also raised. In a similar way, the probabilities of the mutation operators that remove a condition from a transition and change its value are increased if the average score of the complexity aspect is low. These changes on the mutation probabilities allow the IEC method to take into consideration more information about the human evaluation.

The Surrogate Approach

After the human interactors completely evaluate a population, a surrogate function is constructed using the average solution scores in the population. We employed a surrogate model based on the work of (Kim et al. 2014) that uses a Radial Basis Function Network (RBFN). The RBFN can be applied to different problem representations and demands only the existence of a function to compute the distance between two solutions in the solution space. The distance from two C^3 *Cubes* Projects was defined as thin-plate splines using the difference between their graph structures.

As said before, the surrogate mimics the original objective function, which, in our case, resides in the users' mind. Therefore, we can use the recently created surrogate function to evaluate the solutions in the population. For the evolutionary cycle immediately after constructing the function, the surrogate will provide the same results given by the human interactors. Then k more cycles should be performed using only the surrogate function to evaluate the solutions in the population, with k a predefined and fixed number. After the $k - th$ iteration, the resultant population is assigned again to the interactors for evaluation, the surrogate function is reconstructed and the process repeats.

Virtual Environment for Simulation

In order to allow a quick population evaluating, a web-based 3D virtual environment was designed for simulating C^3 *Cubes* projects. The environment allows simultaneously several users to play with and evaluated projects. Each time a user accesses the environment, he or she sees five C^3 Cubes projects randomly taken from the current population (projects that were not evaluated yet have a higher chance of being chosen). The user evaluates the cubes according to the three aspects mentioned before and sends the results to the IEC system (that runs in background and performs $1 + k$ evolutionary cycles when all solutions have received two user evaluations). The user is required to play with every C^3 *Cubes* Project for at least 30 seconds, and to move all three cubes at least once, in order to unlock the evaluation area of the project. These requirements are intended to avoid fake evaluations.

The 3D simulator supports two interaction modes: using a mouse for holding, shaking, moving and rotating the cubes; and using a tangible interface for interacting with the virtual cubes. The tangible interface consists of a Web Camera and three paper cubes of small sizes with fiducial markers at their sides (for an Augmented Reality (AR) application). The paper cubes need to be cut from a sheet and assembled. Each one controls a unique C^3 cube. By moving the paper cubes in front of the Web camera, the user gets the same movement on the cubes at the virtual environment. Figure 2 illustrates the system's functioning with the tangible interface activated.

The tangible interface intents to bring the user a bit closer

Figure 2: Virtual environment for simulating and evaluating C^3 Projects, with the tangible interface activated.

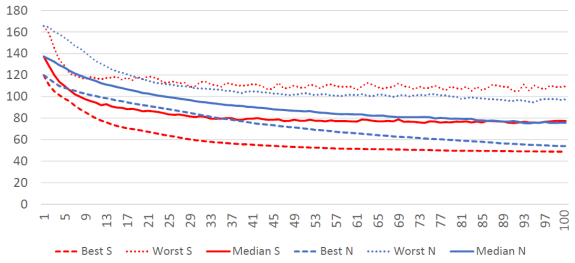

Figure 3: The average fitness value (Y-axis), considering 100 independent trials, of the Best, the Worst and the Median individuals for the 100 generations that actually used the fitness function (X-axis). The Surrogate Evolution is shown in red and the Normal Evolution in blue.

to the interaction with the real C^3 cubes, making the experience more realistic and motivating. The user may opt for using one interface or the other.

Experiments

Two experiments were done for testing our approach. The first experiment aimed at verifying the effectiveness of using a surrogate function. The second experiment focused on identifying whether IEC guides to new interesting behaviors for the C^3 project.

Testing the surrogate

For testing the surrogate approach, we replaced the user evaluation by a simple fitness function (to be minimize) that computes the distance of a solution to a predetermined fixed target[2]. The evolutionary process was implemented and set for running for 100 normal generations (using the fitness function). After every evolutionary cycle of the normal IEC algorithm, we computed k generations using the surrogate function. This results in $100 \times (1 + k)$ generations in total.

We tested two scenarios: with $k = 0$, meaning no usage of the surrogate function, and with $k = 5$, representing five generations using the surrogate for each normal generation. The whole process was repeated 100 times and the average quality of the best, the worse and the median solutions were computed for the generations that employed the distance fitness only.

A chart comparing the results for the surrogate evolution ($k = 5$, referred as "S") against the normal evolution ($k = 0$, abbreviated as "N") is shown in Figure 3, with colored lines. We highlight that the use of the distance function as the fitness is a metaphor for having users performing the evaluation (what can demand a considerable amount of time). The use of the surrogate function, on the other hand, is a fully automatic and relatively fast process.

From the data, we realize that the surrogate evolution has a better early gain in comparison to the normal evolution. By generation 30, the surrogate approach had produced, on average, best solutions that would be surpassed only at generation 70 by the normal evolutionary process. Nevertheless, as more normal generations take place, the gap in quality between the two approaches decreases. After 100 generations, the normal approach tends to provide a better result then the method with the surrogate. This might be because the way the surrogate function models the search space, very often including points with high evaluation values that prevent a fine-grain focus of the search method. This also explains way the red line on the top of the chart (representing the average value of the worst solution in the surrogate approach) is quite erratic and reaches a plateau soon in the computation.

Based on these observations, and considering that, for the IEC, only a small number of generations with user evaluation is possible (in general from 10 to 50 generations), we can see that the use of the surrogate function represents a significant gain over the other approach. In addition, the two approaches can be combined by setting $k >> 0$ at the beginning and moving it to 0 when the change of the quality of the best solution becomes small.

Evolving behaviors

For the second experiment, a complete evolutionary system coupled the Web-based virtual environment for simulation were used. The population size was set to 15 individuals and two evaluations of each individual was required for triggering the evolutionary cycle. The number of automatic surrogate generations, after each complete user evaluation of the population, was $k = 4$. The initial population was composed by four small human-designed C^3 Projects (having two states in each and covering different types of state-change conditions and feedback actions) and eleven randomly generated projects. The three cubes in every initial human-designed C^3 Project had the same state machine.

The experiment involved undergraduate students from a computer graphics discipline at our institution. It was conducted for a week, during which the students had access to the Web application. Each time a user accessed the system, he or she was presented with five C^3 Projects, randomly chosen from the current population, for interaction and evaluation. After evaluating all five solutions, the system would load five new projects (from the same population, or from a new one if an evolutionary cycle was completed).

During the period when the system was active, 11 users participated by conducting more than 210 evaluations as a

[2]The target solution was a behavior manually designed for the C^3, consisting of 6 states and 36 transitions.

group, which led to 6 new generations of the population. The participants could use either the mouse interaction or the tangible interface. The results obtained in this experiment are presented next.

Results The cubes' actions were divided into three classes in order to better analyze them. The first class was named *Light*, and consists of actions that turn the light on or off and make the cube light to blink. The second class, called *Color*, contains the actions that change the cube color. These actions include changing the light color to a random value or to a specific variation from a color palette. The last class was named *Sound*, and has actions that play a sound.

The various metrics that will be explained next are computed in the following way: first, a metric to be analyzed is defined for a cube; Then it is averaged for the three cubes of a C^3 Project; Finally, the grade of a population is computed by averaging the metric for all C^3 Projects within that population. This rule applies to all metrics.

Table 1 summarizes the main metrics for the population in each generation. The rows are: the population identifier (ID); the average number of states per cube (STT); the average number of transitions in each state of each cube (EDG); the average number of conditions inside each transition in the cubes (CND); the average number of actions in each state (ACT); the average number of conditions that use other cube's information, such as if one of the other cubes has its Face 1 up or if the other cubes are nearby (LC); the sum of *Light* actions divided by the total number of states (LTC); the sum of *Color* actions divided by the number of states (CRC); the sum of *Sound* actions divided by the number of states (SDC); the maximum grade a C^3 Project obtained in the population (MXG); the minimum grade a C^3 Project obtained in the population (MIG); the average grade of the C^3 Projects in the population (AG); and the standard deviation of the C^3 Projects' grades of the population (SDG).

Table 1: Main metrics of the population in each generation.

ID	0	1	2	3	4	5	6
STT	1.7	1.9	2.1	2.1	2.2	2.6	2.7
EDG	1.5	1.9	2.0	2.3	2.3	2.9	4.1
CND	3.0	1.5	1.2	1.5	1.3	1.2	1.1
ACT	1.9	1.5	1.8	2.8	3.3	3.7	3.7
LC	1.7	1.2	1.1	1.2	1.0	0.8	0.8
LTC	1.1	0.7	0.7	1.1	1.5	1.7	1.7
CRC	0.7	0.2	0.4	1.0	1.2	1.5	1.6
SDC	0.3	0.6	0.7	1.9	1.0	1.0	1.0
MXG	8.5	8.5	10.0	10.0	9.3	10.0	-
MIG	2.0	2.0	1.8	1.0	1.8	1.0	-
AG	5.9	5.1	5.7	5.8	5.8	6.2	-
SDG	3.3	2.9	3.2	3.5	3.2	3.2	-

The proposed approach was able to evolve the C^3 Projects, obtaining behaviors different from the initial ones. The complexity of the state machines in terms of number of states and transitions increased with the generations and was possible to identify changes on the transitions and on the state actions in terms of type and other parameters.

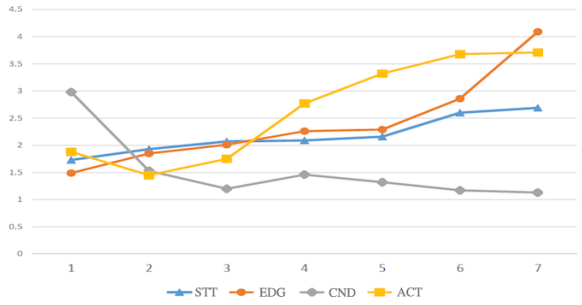

Figure 4: Average number of states (STT), edges (EDG), conditions (CND) and actions (ACT) per generations.

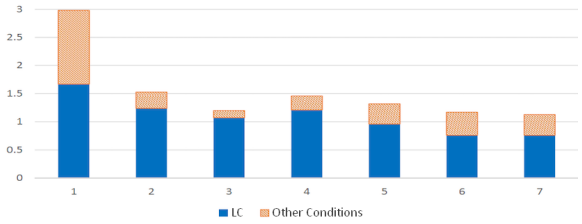

Figure 5: The average number of conditions that use the status of other cubes (LC) versus the average number of conditions that use only the cube's own status (Other conditions).

The line chart in Figure 4 shows the changes of some metrics that measures the complexity of the C^3 projects in the population over generations. There is an initial decrease of the numbers CND and ACT. This may be because of the initial random projects, which contain many non intuitive interactions that are badly evaluated by the user. After one or two generations, however, the solutions started to become more suitable to the users' interest and the measures increased or became stable.

Figure 5 highlights another pattern. The average number of conditions based on the other cubes' information (LC) and the average number of the remaining conditions are very high and almost proportional in the first population, due to the randomized initialization process. Then, they reduce significantly and start to become more stable with a higher percentage for LC. The increase in the proportion of LC is a favorable factor, since it indicates that there is relatively more interaction between the cubes in a C^3 project.

The line chart in Figure 6 shows that the normalized number of Light and Color actions (LTC and CRC) had an initial decrease, as happened with some other metrics. After a few generations, the normalized number of *Sound* actions (SDC) grew and found a limit; the values LTC and CRC kept growing as their limits were higher (with more types of actions available). The increase in these numbers are also favorable in general, because it possibly indicates more feedback to the interactors. However, if they raise too much, the cubes' behaviors may be very similar, with all interactions possibly resulting in the same type of feedback.

It is useful to understand the best evaluated C^3 Project in the last population, as well. That solution presented more

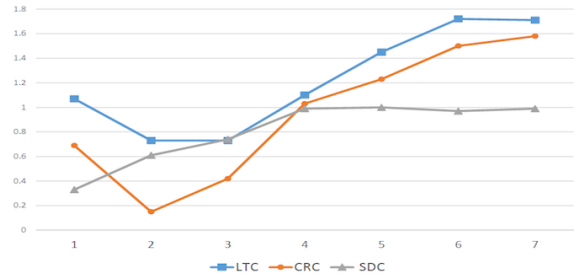

Figure 6: Average of *Lights*, *Color*, and *Sound* actions as LTC, CRC and SDC respectively, per generation.

complex and interesting behaviors than that in the initial C^3 Projects. The three cubes in final best solution also had distinct state machines.

Finally, although we offered a tangible interface and considered that it would be more appealing for playing with the cubes, most users preferred the mouse-based interface. One reason for this may be the extra effort for printing, cutting and assembling the paper cubes. Furthermore, it was necessary to use the mouse anyway for giving the grades.

Conclusion

This paper proposes the integration of computer processing power and human evaluation by means of an interactive evolutionary computation approach. The approach is assisted by a surrogate function and a simulation environment for the creation of new C^3 *Cubes* projects.

The human-computer combination provides new and more interesting projects as the evolution of populations goes on. The final projects (or just the best evaluated one) may be implemented and uploaded to the Arduino board in real cubes, without the need of programming skills or knowledge about the hardware and the software of the artistic project. There is still the need of time for playing and evaluating several C^3 projects, but this is significantly reduced by using of the simulation environment and the surrogate function.

The application as a virtual environment still lacks some physical aspects, which unfortunately is required to provide a better experience. Thus, we are planning to build a simultaneous experience with a physical C^3 *Cubes* project installation being dynamically loaded with the best current solution in order to allow the users to interact to it and to add his or her evaluation to the population.

Further studies in this field include the adaptation of techniques to compensate the surrogate long-term loss. Also, the use of a different rating approach, in which more information can be inferred from the evaluation of C^3 *Cubes* projects, could help the evolutionary process. Finally, we intend to create incentives for the users to use the tangible interface over the mouse, and to extend the investigation of the benefits of these interfaces.

We also consider extending the our proposal to game design, for example, for creating scenarios, landscapes, racing roads and new RPG avatars, which present the same challenges described in the present work.

Acknowledgments

The authors would like to thank FAPEG (Fundação de Amparo à Pesquisa do Estado de Goiás), process number 01/2018, for the financial support.

References

Draves, S. 2005. The electric sheep screen-saver: A case study in aesthetic evolution. In *Workshops on Applications of Evolutionary Computation*, 458–467. Springer.

Hastings, E. J.; Guha, R. K.; and Stanley, K. O. 2009. Evolving content in the galactic arms race video game. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, 241–248. IEEE.

Kim, Y.-H.; Moraglio, A.; Kattan, A.; and Yoon, Y. 2014. Geometric generalisation of surrogate model-based optimisation to combinatorial and program spaces. *Mathematical Problems in Engineering* 2014.

Lages, W. S.; Gobira, P.; and Marinho, F. 2017. Better hands. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, C&C '17, 453–455. New York, NY, USA: ACM.

Madera, Q.; Castillo, O.; Garcia, M.; and Mancilla, A. 2016. Interactive evolutionary computation with adaptive mutation for increasing the effectiveness of advertisement texts. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, 1–6. IEEE.

Makiwan, D.; Yoshida, K.; and Koppen, M. 2017. Interactive evolutionary computation of color palette design enhanced by impression words. In *Platform Technology and Service (PlatCon), 2017 International Conference on*, 1–6. IEEE.

Röggla, T.; Wang, C.; Perez Romero, L.; Jansen, J.; and Cesar, P. 2017. Tangible air: An interactive installation for visualising audience engagement. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, C&C '17, 263–265. New York, NY, USA: ACM.

Romero, J. J., and Machado, P. 2007. *The art of artificial evolution: a handbook on evolutionary art and music*. Springer Science & Business Media.

Semet, Y. 2002. Interactive evolutionary computation: a survey of existing theory. *University of Illinois*.

Sims, K. 1991. *Artificial evolution for computer graphics*, volume 25. ACM.

Takagi, H. 2001. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE* 89(9):1275–1296.