Gesture-based multi-threaded programming interface: A prototype

Interface basada en gestos para programación multi-hilos: Un prototipo

Raúl A. Herrera-Acuña^{1*}

Recibido 4 de Marzo de 2019, aceptado 25 de Agosto de 2021

Received: March 4, 2019 Accepted: August 25, 2021

ABSTRACT

Software programming for concurrent execution is not a simple task. The main issue related to this work is the lack of visibility of the multiple and parallel execution over the threads. Multi-core processor technology is a reality nowadays, and the only way to use modern hardware at its full capacity lies in the use of concurrent software. In this paper, a new 3D framework is presented, based on hand gestures that manipulate multiple threads of execution and deal with the visibility issues, using a 3D programming environment, including a set of experiments to evaluate the methodology's performance.

Keywords: Hand gesture interaction, 3D programming HCI, multithreaded programming.

RESUMEN

La programación de software para ejecución concurrente, no es una tarea sencilla. El problema principal relativo a este trabajo tiene que ver con la falta de visibilidad de las ejecuciones múltiples y paralelas en los diferentes hilos. La tecnología de procesadores con núcleos múltiples es una realidad hoy en día y la única forma de utilizar hardware actual a su plena capacidad radica en el uso de software concurrente. En este artículo, se presenta un nuevo framework en 3D, basado en el uso de gestos de manos para manipular múltiples hilos de ejecución y lidiar con los problemas de visibilidad, usando un ambiente de programación 3D, incluyendo un set de experimentos para evaluar el desempeño de la metodología empleada.

Palabras Clave: Interacción por gesto de manos, programación IHC 3D, programación multi-hilo.

INTRODUCTION

Attempts to provide a bridge between the real environment and the computer interfaces have become an important topic in human-computer interaction research. The significance of these Human-Computer interaction advances have been addressed by several researchers in the last decade, highlighting the importance of creating new communication methods between humans and computers, replacing traditional methods and devices [1].

Video game technologies are leading the way in generating more natural interactions using the body commands of users, especially hand gesture-based interactions. These developments are due to the flexibility and intuitive use of the hands during interactions and the manipulation of 3D objects in a 3D space, like any daily based human activity. The importance of providing 3D spaces and interactions lies in the fact that simply having two-dimensional interactions is not enough to perform specific tasks naturally, especially when

¹ Universidad de Tarapacá. Facultad de Ingeniería. Arica, Chile. E-mail: rherrera@uta.cl

* Autor de correspondencia: rherrera@uta.cl

these activities are performed in three dimensions in the real world. Advances in in-depth capturing devices have provided novel approaches to interface systems, as Microsoft Kinect has shown lately [2], which addresses the previously mentioned issue.

A novel research area for 3D hand gesture interaction is multi-thread programming since the representation of multiple lines of code working simultaneously can be better represented and understood in a three-dimensional graphic environment than with simple sequential code because of the resemblance with the real environment and the possibility of multiple views [3 and 4]. The complexity of generating applications using multiple threads generally relays in the lack of representation of the final program and its operation, which cannot be seen under a 2D representation. The issues related to the working environments in developing applications for multiple processors/threads are not new, and there have been advances showing the need for novel and advanced interactive mechanisms [5, 6 and 7]. Using graphic icons to represent data elements and functions helps to clarify their purpose in programming, and environments have successfully represented these tools in a reasonably accurate and intuitive way [8]. Most of them lack 3D representation, which, as indicated previously, increases the understanding of the encapsulated information and the productivity during the utilization of multiple sources of information, especially in complex tasks such as multi-thread programming [9].

This paper presents a novel approach for generating multi-thread code using hand gesture interactions in a 3D environment, introducing the concept of multidimensional software programming and design. Among the other advantages of 3D software development, the proposed framework allows the user to navigate a more human-friendly code development environment. In contrast, the proposed human-computer interaction mechanism takes advantage of all the features and concepts of 3D interaction systems. The following section presents an analysis of previous work, showing the progress in the related areas. In the following section, an analysis of previous work is presented, showing the progress in the related areas. Then, the proposed environment is analyzed, providing details of a novel interaction framework for software development. Finally, an evaluation methodology of

the proposed interaction approach is discussed, and conclusions are presented. The contributions related to this paper are the definition of an interaction interface for multi-threaded programming, a 3D novel representation of iconic tools, and the introduction of a 3D representation for multi-thread programming.

PREVIOUS WORK

The work of MIT's Tangible Media Group [10] presents an alternative to replacing the text-driven systems in geographic information systems (GIS). Their approach allows direct interaction with geographical data. The user can modify and analyze surfaces as part of the interface using tangible objects (such as blocks, trees, hills, and others.) integrated with augmented reality environments, depicting changes of specific terrain characteristics. A 3D display allows the user to visualize the work in progress, using the "tangible bits" paradigm and digital elevation models of a surface. The Tangible Bits paradigm [11] consists of an augmented reality system combined with an intelligent environment, where users can manipulate real objects on a surface and obtain feedback from the interaction surface based on digital projection. The system uses laser-based technology to detect the movements of the user and advanced image processing software based on augmented reality techniques to generate feedback. The main problem of implementing this kind of interface is the high cost of the devices used and the difficulty of configuring all the hardware and software for a single application. The interaction mechanism was also limited to the particular problem without providing any flexibility. These types of advances show the possibility of using 3D interaction scenarios in other kinds of information manipulation, such as programming.

Finding an adequate way to design a system with 3D interaction and a natural interface is a complex problem because of the lack of information regarding the working area and the user's needs. Paradigms like case-based reasoning and the use of support frameworks to design new object-oriented architectures presented in the work of Vazquez [12] can help overcome the previously mentioned difficulties. The importance of taking into account previous designs and using them in materializing new software is discussed. Based on this approach, the test system can provide "advice" to developers

on the choice of architectural software components. These pieces of advice are based on evaluating a set of quality aspects, for example, performance, modifiability, or scope. As a consequence, experience is required to decide if the components are really suitable for software design.

The possibility of representing in 3D a large amount of information was explored exhaustively in the work of Marcus [13], where 3D models were used to represent a software system allowing a better understanding of high dimensional data. The most relevant aspect of this system is related to user interaction and 3D visualization, showing multiple nested levels of the code-based on colors and viewing models. Probably the major drawback of this design is that the interaction is in 2D and that it is based on traditional interaction devices. Consequently, it retains the disadvantages of a 2D interaction in a 3D environment, with the interaction mechanisms limited with this representation model.

There have been many attempts to achieve a pure 3D programming language over the years. A relatively modern example is the Solid Agents in Motion –SAM– [14], a visual 3D programming language for parallel systems and animation. The language is based on agents (3D objects with an

arbitrary number of input and output ports) that interact by exchanging messages (a data structure that can be an identifier, a value, identifiers of the sender or receiver enclosed as text). The behavior of each agent is specified by production rules with a condition and a sequence of associated actions, as in-state machines. The graphical representation of each element is initially a semi-transparent 3D model, such as cylinders, spheres, or cones. Agents and messages have an abstract and a solid 3D shape, where the abstract concept consists of a description text based on the agent's action environment, the agent itself, and the production rules applicable to the agent.

In contrast, the solid 3D representation corresponds to a graphic 3D model of the agent. The interaction with these elements (mouse and keyboard-based interaction) can be achieved by moving over the 3D object and getting more specific information by double-clicking over the object. Each 3D element has several connection ports for data input and output, depending on the definition and function of the agent. The agents use these ports to send and retrieve messages, and each port has textual identifiers and colors to indicate their function. In Figure 1, an example of an agent can be seen. The execution of the programs based on these graphic

Source: [14]

Figure 1. SAM's agent architecture.

agents is based on synchronous communication in a cycle of two phases: i) agent execution and ii) agent communication. In the first step, all the agents check their execution rules, perform their tasks in the respective order and then pass the message to the next agent, according to their execution rules. Even when this model of programming presents possible advantages, the complexity of the rule generation and the trivial non-natural methods of interaction (using mouse clicking) present a poor use of 3D interaction capabilities, especially in the aspect of visualization and interaction with the programming environment.

Later on, attempts on 3D graphic interaction proposed animated execution of programs instead of text-based debugging. One example of this is the approach presented by 3D-PP, a visual programming system with 3D representation [15], where the construction of programs is based on a hierarchical graph of nodes and edges, where the nodes correspond to data (represented by spheres), operators (represented by inverted cones) and process (represented by pillars). Each process is defined by a set of rules, composed of a condition (used to select one rule from multiple choices at runtime) and a body (that defines the performance rule following the state machine model). All the components can be accessed and modified using direct input devices, such as a mouse or any pointing mechanism. Once the program starts, there are multiple options, including stopping the animation and modifying the program to solve possible problems and bugs. The main issue of this implementation lies in the fact that all the graphical construction elements work as a normal 2D code-based program, where the process of execution and visualization needs to be defined step by step. Also, the interaction is still based on traditional interfaces, not using the available features and advantages of 3D interaction and manipulation, such as rotations, multiple angle view of the software, zooming, and others.

In [16], freehand gestural target selection usability with different 3D marking menu layouts and target directions were analyzed. Gestures comprised a standard library available as default, offering a comprehensive way to integrate different kinds of multi-touch and direct input interaction devices. More details on the current generation of 3D user interface (3DUI) applications and their development

issues are presented in the survey in [17]. Criteria for measuring the development difficulty of 3DUI and two benchmarking 3DUI toolkits were also suggested.

The possibility of improving the speed of performing complicated tasks focusing on their parallelization has been explored during the latest years. Nowadays, multi-thread programming is necessary to use the full capacity of the available multi-core processors with built-in capabilities to perform concurrent tasks. However, the main problem with threads is not the threads themselves but the lack of visualization of related components during the design and programming processes that cause several problems related to concurrency.

Another issue related to multi-threading is the lack of standardization of the programming models for parallel computing in heterogeneous systems. The MERGE framework presented by Linderman [18] addresses this problem by considering an intense use of “libraries” to deal with tasks and the data distribution between the different components of heterogeneous systems, using a unified programming approach instead of the classic static/dynamic compilation method. This dynamic approach allows the programmer to designate specific tasks to architectures without knowing exactly which machine is going to be carrying out each task. The framework uses knowledge of the architecture (based on a set of libraries) to distribute the work between the components of the system. This approach addresses several issues related to parallel programming in multiple machines. Nevertheless, it is not a real solution when parallel computing is done in just one of the processors and task assignment is defined manually, which generates the lack of visibility of the different components in the application. Under this perspective, using a graphics-based environment becomes more suitable to solve programming issues of multi-core processors.

The approach for multi-threads presented by Harrow [19] aimed at the need of visualization of concurrent executions and provides an approach to model the working threads in real-time and their progress in the system, but still used as a starting point written code, where learning time is higher, and errors are more probable because of the complexity of the elements to understand, learn and use. Also,

it does not provide multiple view options because the threads' tasks overlap. The availability of a 3D graphic framework to visualize how the program is constructed becomes more desirable and intuitive.

Combining a 3D visual interface and a 3D gesture-based interaction system seems an attractive and interesting way to solve the problems previously discussed regarding concurrent programming. The proposed approach of a novel framework is presented in the following section.

METHODOLOGY FOR MULTI-THREADED 3D PROGRAMMING

The proposed multi-thread framework is divided into key elements related mainly to interaction capabilities and interface mechanisms.

At this stage, it is necessary to explain the importance of using this approach in a programming environment before analyzing the framework. The use of 3D interaction is possibly the most arguable issue in this work, but there are several reasons related to previous works in the area of human-computer interaction that support this approach. In more detail, there are several studies on designing systems that can support 3D interactions, providing users more confidence and comfort during the interaction process. The advances in infrared motion capturing devices, which allow interacting directly with systems using bare hands, have been used successfully to achieve more natural interaction. These devices, which have been extensively used in entertainment, are gradually entering other research and development areas, especially those related to hand gesture-based interaction frameworks. These actions and gestures can be easily applied to interfaces related to graphics-based programming [20 and 21]. The idea to use 3D metaphors to represent a program is not new, especially in robot programming, where several tasks are performed in real environments, and an iconic visualization is utilized to simplify the definition of graphic elements for specific tasks [22]. These ideas are also applicable to multi-objective linear programming (i.e., programming algorithms to solve optimization problems for multiple variables that can use parallel programming to solve the mentioned problems [21 and 23]) and 3D programming for dynamic systems (i.e., systems that change during time, such as particle movements, bioreactors,

communication systems, and others.). Consequently, we might argue that there is an interesting possibility to use the proposed methodology for multi-thread programming.

Interaction definition for a 3d gesture-based programming environment.

The system interaction is based on direct hand instead of hand and fingers gestures. In this approach, the skeleton tracking and palm 3D detection provided by the Microsoft Kinect SDK are used to perform the interactions.

Since the interaction with the system is directly based on 3D hand gestures and using 3D objects as metaphors, three basic interactions are defined:

- Rotation: Rotations in the system are based on simply moving the hand in the defined working area. This action only works in the workspace of the framework, and its main utility is to shift between the different available threads of the program. The rotation of the threads is around the vertical axis, moving from left to right or vice versa with a predefined 3D space assigned to each thread. The action is performed by sliding the hand over the working area, making it possible to perform it either from left to right or from right to left. It is also possible to perform it only if no other actions or gestures are performed at that time by the user.
- Grab: The process of grabbing elements is necessary to select and add programming elements in a given thread. These elements are in the programming tools area of the screen (see Figure 2), outside the working area. The grabbing process is performed in two steps: first, placing the cursor over the item to be selected and then, moving the element to confirm that the process was completed (after that, the 3D element will move following the hand indicator). The selected element will float following the hand position allowing the user to place it in the working area and add it to the program at the necessary location.
- Release: The release process is performed oppositely and assumes that the system is grabbing. It consists of an initial stage where the user's palm hovers over the desired end location and a second stage where the user pushes the element towards the screen in 3D space to release it. Also, at that moment, the

system can give the option to access features of the given element and “unlock” other actions once the process is finished, such as rotations or grabbing a new element.

- Release incorrect item: The process to release an incorrect grabbed item must be performed in the same way as the standard release but in a different area of the screen, which will allow the user to detach the incorrectly selected item from the hand indicator and be able to pick a correct item.

MULTITHREAD INTERFACING FRAMEWORK ANALYSIS

The gesture definitions given above are used to ‘construct’ a program, but it is also necessary to clarify some specific elements related to the working environment of the proposed framework.

Metaphor analysis

Metaphors are essential to represent as much as possible real components based on 3D elements. The hand gestures analyzed previously, and the graphic icons must facilitate the tasks and needs of a programmer instead of making it more difficult. This ease of tasks can be achieved by understanding the application area and automatically adapting the

tools [24]. In our approach, the proposed framework’s model developed is shown in Figure 2. This figure represents a view of the main interface window with the basic working elements.

As seen in Figure 2, the “programming area” is in the center of the interface, with the programming elements (3D models that represent typical programming structures) on the left and right of the interface. Also, in the case of our experiments, an “example” area is added at the bottom left. Also, at the upper part of the working area, there is a “release element point,” which allows the developer to release an undesired object previously selected. The elements and the interaction will be analyzed in more detail in the following sections.

Main 3D Workspace

The main 3D workspace was presented previously and corresponded to the area where the program will be created, and this space is further divided according to the number of the supported threads. Four threads are utilized in the proposed interface prototype; therefore, the space is divided equally into four subareas, as shown in Figure 3.

As seen, the workspace is divided equally, providing a specific working area to each thread, where

Figure 2. Programming graphic interface, where (1) corresponds to the interaction window; (2) corresponds to the programming area; (3) corresponds to the programming element’s placement area; (4) corresponds to the area for the programming tools (5) corresponds to the specific space for each tool and (6) corresponds to the element release area.

Figure 3. Workspace for four threads.

the graphic elements can be inserted to develop the application. This place is where the rotation and release actions can take place. It needs to be mentioned that there is only one thread active at a given time and to move to the next one; the rotation gesture needs to be performed. The previous configuration allows the development of software in parallel.

3D Iconic Tools

The 3D iconic tools are the 3D models that represent basic programming elements such as conditions, variables, mathematical operators, or other functions. The user can grab these tools to construct the program that will run in the selected thread.

APPLICATION DEVELOPMENT PROCESS

The presented 3D programming model aims to solve the problem of the lack of visibility in the multi-thread programs, limiting the developer's possibilities to see clearly how the threads connect with each other. Given the previous problem, a 3D iconic representation can be the solution, allowing different execution threads simultaneous visibility. Also, a 3D iconic metaphor, as it was discussed previously, increases the understanding of the problem and improves the suggested solutions.

Developing a program can be fairly simple, and an example is provided to demonstrate the whole procedure. The initial step of the process is to grab and move the first icon in the workplace. Adding elements to each thread is similar, grabbing and

moving them in the work area. After completing the first thread, the user just needs to rotate the workplace to continue with the following thread until all the tasks in all the threads are completed.

ANALYSIS OF THE EVALUATION PROCESS

The evaluation process of the proposed framework is based on a simulated example that involves tasks related to the development of a specific multi-thread program. The proposed experiment provides information about two specific areas: Performance and user satisfaction, allowing both qualitative and quantitative evaluations. Additionally, a comparative study with the traditional development interfaces for multi-thread applications is presented.

This experiment needs to provide mechanisms to evaluate all the features of the proposed framework, and it is necessary to evaluate them using an example, simple enough to be understood by people who either do or do not have any experience in multi-thread programming. Based on these requirements, the task of adding an array of numbers using 4 threads was selected, and the interface that was used is the one presented in the previous section. The C++ code for that task is presented in Figure 4. The experiment provides a set of tools necessary to program the code in each thread. The initialization of values and variables is not part of this experiment.

The experiment follows the stages presented below:

- The presentation stage (where the interface is presented and explained to the users).
- The practice stage (where the users can interact with the interface and use the available features), and.
- The task execution stage (where the users perform the task and quantitative factors are registered, such as task execution time and quantity of errors).

After that, the users are asked to complete a questionnaire with similar questions to the questionnaires provided by IBM's research about new interfaces [25].

The main elements related to the experiment are analyzed in the following sections.

<pre> 1 //Global Variables 2 LL = 1000000 3 REPS = 1; 4 global_sum=0.0; 5 aa[LL] 6 mutex_sum; 7 8 // Function that adds the elements of an array 9 void sum 10 { 11 i=TID*(N/NT), i1= i0 + (N/NT); 12 localSum=0.0; 13 for (r=0; r<REPS; r++) 14 for (i=i0; i<i1; i++) 15 localSum += aa[i]; 16 17 mutex_sum.lock(); 18 global_sum = global_sum + localSum; 19 mutex_sum.unlock(); 20 } 21 22 23 24 </pre>	<pre> 25 main() 26 { 27 // Initialization 28 for (t=0; t<LL; t++) 29 { 30 aa[t] = t+1; 31 } 32 // Create 4 Threads 33 thread_vector = new thread [4]; 34 NT = 4; 35 for (t=0; t<NT; t++) 36 { 37 thread_vector[t] = thread(sum, t, LL, NT); 38 } 39 40 //Join the Threads 41 for (t=0; t<NT; t++) 42 { 43 thrs[t].join; 44 } 45 46 //Display the result 47 print global_sum 48 } </pre>
--	--

Figure 4. C++ code for multi-threaded framework's test.

Set of gestures

The set of gestures selected covers the interaction capabilities described in section.3.1. The set of gestures is defined as follows:

- Rotation: The gesture of rotation is to move the left hand from left to right. The rotation is restricted to just one direction to avoid problems understanding the user's interaction (from left to right), and it can be executed with just one arm. For our experiments, the left arm was selected.
- Grab: To grab elements, the user needs to place the hand over an object and push. At that moment, the object will be "attached" to the hand indicator and move along with it. This gesture can be performed with both hands and allows grabbing two 3D objects simultaneously.
- Release: The process to release an element depends on the selected object. If the selected object corresponds to the desired element, it must be placed over the special contact points (programming slots). If the object is not the right one and the user does not desire it, it can be released by placing it over the drop section and selecting another one.

For each of these actions, a set of thresholds is defined. These thresholds were selected experimentally. Figure 5 helps to explain how the thresholds were defined.

Figure 5 represents the interaction area on the screen (graphic interface) with height Y and width

X, where the toolboxes have a height equal to C and width equal to D (using both approximately 30% of the interaction area). The programming area with height A and width B (using approximately 24% of the interaction area) and the release area (red square on top) of width and height E (using the 3% of the interaction area). All the areas where the programming elements can be "grabbed from" or "placed in" are square with size E. R represents the length of the movement needed to rotate the programming area moving from one thread to the next one, which can be performed in any place of the interaction area.

Task Description

The task performed by the users consists of programming all four threads to execute the addition of several elements in parallel. The required 3D elements to complete the task are the 'loop cycle' graphic icon, which will "contain" the variable (i.e., accumulated value) and the summation process of the consecutive values. The first element to be selected (grabbing) is the loop, then the variable, and finally the addition. There is no specific sequence in the selecting and placing of the elements, and two 3D objects can be added simultaneously. To clarify the task, Figure 6 presents the sequence of an ideal execution of the task, but any approach is correct if the final outcome is the same. The same process presented must be repeated 4 times once for each thread until the whole task is completed.

Figure 5. Interaction areas for the experimental interface, where the external light blue square represents the whole interaction area (with measures $X \times Y$); the central rectangle represents the programming area ($A \times B$) where each grey square represents the areas to place a programming 3D icon ($E \times E$). The rectangles in the sides represent the toolboxes ($C \times D$) where the blue squares represent the areas to grab the programming icon tools ($E \times E$); the red square is the wrong element release area ($E \times E$); and R represents the range of movement for rotation, that can be performed anywhere in the interaction area.

Figure 6. Interaction sequence to create a thread in the interface a) First step: Grab two elements. b) Place them in the work area c) Grab the third element d) Place the element in the corresponding position e) Rotate the work area to start a new thread.

In this example, the expected solution is provided to the users in the example box to help them with this task. In the end, the interface provides the required time to complete the whole process. In the following section, the experiment process and the evaluation procedure are presented.

Evaluation Procedure

During the evaluation process the experiments were divided into several steps

Present and explain the experiment and its objectives:

This section aims to provide information about the possibility of using 3D hand gesture interfaces instead of the traditional code-based interaction to develop multi-threaded software. As a result, the proposed experiment performs a comparative study evaluating the users' performance on the proposed prototype interface versus the traditional approach. Since users are informed about the experiment's overall concept and objectives, we move to the next step.

Demonstrate:

The demonstration section aims to present the interface and its elements to the users, answering any related questions. Also, if the users are not familiar with multi-thread programming, a short explanation and examples are given.

Familiarize the subject with the interface:

During the familiarization stage with the interface, the interaction mechanism is presented to the users allowing them to practice basic movements and the on-screen features.

Subject performs the available actions:

The available functions (e.g., rotation, grabbing, among others.) are explained to the users and demonstrated in real-time during this step. Furthermore, they are encouraged to practice and

perform these functions by themselves. The total time of training is a few minutes, indicating how much intuitive the proposed approach is.

Full task Performance:

Since the users are familiar with the environment and with the mechanisms to perform the available functions, the full task that was initially introduced is performed counting the required time to complete it successfully, the number of errors during the users' interactions, and the system errors due to erroneous action detection. The errors were considered according to the gestures described above; therefore, the metrics to evaluate the interface were time to complete the task, number of user errors, and number of system errors; three types of errors were considered: Grabbing, placement of the object, and rotation of the workspace.

Questionnaire completion:

After the successful performance of the task, a questionnaire is completed by the users, evaluating and comparing the available interfaces (i.e., visual 3D and code-based programming). The questionnaire was the tool to collect the user's feedback and to provide a qualitative analysis. During this process, any questions from the users are answered to make sure everything is clear to them. There is no time limit for this task.

The questionnaire model used was based on the questionnaires provided by IBM in their research about new interfaces on usability tests.

The questionnaire is separated into two main sections to evaluate the user experience with the interface, shown in Table 1. In all the questions, the users provide a number from 1 to 5 to evaluate the interface according to the question, where 1 is the lowest score (extremely negative evaluation), and 5 is the highest (extremely positive evaluation).

Table 1. Usability questionnaire for sections 1 and 2.

Section 1	Section 2 - How would you rate:
Q1: Was the interaction easy to understand?	Q4: The Interface?
	Q5: The Performance?
Q2: Was it easy to manipulate?	Q6: The functionality?
	Q7: The objective achieved?
Q3: Is the navigation system intuitive?	Q8: The user experience?
	Q9: The hand gestures selected?

At the end of the questionnaire, a last question is asked about the users' preference over the 3D visual gesture interface and the traditional code programming, where the users select their preferred approach (1 to the preferred interface and zero to the other). The obtained results are presented in the following section.

EXPERIMENTAL RESULTS

Experiments were conducted using 29 subjects aged 20 to 50 years to evaluate the proposed interfaces. Regarding the subjects, 63% were males, and 37% were female. Also, the level of programming knowledge and experience was well distributed among them from novice to expert. The result of the experiments will be separated into qualitative and quantitative analyses in both sections below. The results are shown in terms of medians and median absolute deviations to avoid the influence of outliers.

The statistical validation of the data obtained for the set of experiments conducted was performed using the Wilcoxon signed/rank test [26]. Our data cannot be considered normally distributed, given the level of skewness. However, in this case, given that we have just a single experiment, the data obtained were compared with mean response values and a significance level of 0.01.

Qualitative results

In this section, the presented results correspond to the answers given by the users in the questionnaire previously discussed. The evaluation of the interface from the users is presented in the following tables and graphs. Table 2 shows the median values for the questions answered, including the median absolute deviation over these values. In the first section, the

Table 2. Median values and median absolute deviation in each question for all the interfaces.

Section 1	Section 2
Q1: 5.0 (0.0)	Q4: 5.0 (0.0)
	Q5: 4.0 (1.0)
Q2: 4.0 (1.0)	Q6: 4.0 (1.0)
	Q7: 5.0 (0.0)
Q3: 5.0 (0.0)	Q8: 5.0 (0.0)
	Q9: 5.0 (0.0)

best-evaluated aspect by the users was the simplicity in understanding interaction with the interface. The first section has high values, indicating that the users found the interface intuitive and easy to use. For section 2, the evaluation is positive as well, with significantly high values for the interface, the achievement of the objective, and the selected gestures, indicating users' satisfaction with the interface and the system's overall performance.

Regarding the obtained values during the evaluation, all the aspects presented were ranked above 3.5, indicating the general acceptance of the new approaches based on hand gesture interaction.

Table 3 shows the results regarding the last qualitative question: The users' preference between the hand gesture interface and the traditional text-based interface to work with multiple threads.

Most users generally prefer the hand gesture-based approach over the text-based interface because of the visual representation, which improves the understanding of the given task.

The answers for the two sections versus different age ranges regarding the qualitative analysis are shown in Figure 7 and Figure 8. The standard deviation is added to the plots.

Figure 7 indicates that all subjects evaluated the interface itself with a value over 4 (very good) with a low standard deviation, showing that the evaluation given by the users is, in general, similar, which demonstrates a clear acceptance of the interface. The groups that evaluated the interface with the highest scores were the older users in the 36-40 age range, followed by users between 46 and 50, and users between 26 and 30 years of age. These scores can be explained by the intuitiveness of the interaction process that resembles other modern interaction interfaces (to the younger users) and provides a more straightforward manipulation model (to the older users).

Table 3. User's preference of interface.

Value	Gesture Based	Text Based
Median	1.0	0.0
Median Absolute deviation	0.0	0.0

Figure 7. Evaluation values of section 1 of the questionnaire vs. age range (with median absolute deviation). The bars represent the scores given by a different group of users (according to their ages).

In this case, the statistical evaluation was performed with a mean response value of 3.5 (which indicates a positive evaluation of the interface). The values obtained were Wilcoxon Statistic = 6, $p > 0.01$, indicating a highly significant mean difference from the “neutral” response of 3.5 (one-tailed), indicating the difference is highly positively significant for our approach.

Section 2 presents similar results (Figure 8), but the group of users between 25 and 30 years evaluated the interface better. This result can be explained by the graphic definition of elements that are easier to understand and the distribution of the graphic elements, which provides the necessary space to interact properly.

The statistical evaluation was again performed with a mean response value of 3.5. In this case, Wilcoxon Statistic = 14, $p > 0.01$ (one-tailed), indicating a highly significant mean difference from the response of value, indicating the difference is highly positively significant for our approach.

Quantitative results

This section analyzes, the results obtained by measuring the completion time, the number of user errors, and the number of errors from the interface.

Times:

A summary of the average times to perform the task is shown in Table 4.

Figure 8. Evaluation values of section 2 of the questionnaire vs. age range (with median absolute deviation). The bars represent the scores given by a different group of users (according to their ages).

Table 4. Times to performance for males and females, according to their average age.

Value	Male	Female
Median Age (years)	32.0	32.5
Median Time (seconds)	90.0	92.7

As seen, the median age for males and females is almost the same, but there is a slight difference in the performance time in favor of the male subjects in the experiment.

The time required to complete the tasks is further analyzed, providing a more detailed quantitative evaluation. The figures presented below show how different aspects are related to the speed and the time required to accomplish the tasks.

Figure 9, shows that the best performance corresponds to the subjects between 20 and 35 years, with relatively low median absolute deviation. The subjects over that age present faster results (less time to perform the complete task). The slowest performance was obtained by the group of users between 41 and 45 years old; however, the median absolute deviation was higher for that group. That fact can be related to the performance speed of the different gestures involved in the task, especially the grabbing and placement tasks.

Figure 9. Median performance time of subjects based on their ages (with median absolute deviation). The bars represent the time to perform the task given to different groups of users (according to their ages).

The female subjects (Figure 10) present their best performance (less than 100 seconds) in the range of 20 to 29 years of age, with a low deviation. The slowest performance was for the female subjects in the age range of 40 to 49, with times near 200 seconds, but with high median absolute deviation, indicating that the performance of the subjects in that age range varied probably because of the lack of understanding of the gestures or the speed to perform the grabbing and placing (as it was observed during the experiments, some users performed the displacement of the 3D icons slower compared with others). The case of male subjects is similar. As in the case of the female subjects, the male users with best times belong to the age range between 20 and 29 years old and the slowest results were obtained by the users between 40 and 49 years old. However, the times in the slowest case are better than in the female case and the standard deviation is a lot lower, indicating the users' performance, especially on grabbing and placing the 3D icons, was significantly faster and consistent for all male users.

The statistical evaluation, in this case, was performed against the mean value of the time to perform the given task; 93.1 seconds. The values obtained were Wilcoxon Statistic = 134, $p > 0.01$ (two-tailed), indicating that there is no statistically significant

Figure 10. Performance time of male and female subjects based on their ages (with median absolute deviation). The blue bars represent the time to perform the task by male users, and the purple bars represent the time to perform the task by female users, given different groups of users (according to their ages).

evidence to affirm that the time results are more than the average value of the time performance for our approach.

User Errors:

The user errors are summarized in Table 5.

The table above shows the average results for the 3 types of user errors: Rotating the working area, grabbing the 3D programming elements, and placing them in the correct locations. As it can be observed, female subjects had better results than male subjects. Also, the highest value of error was in the process of grabbing an element. In Figure 11, the user errors over different age ranges are summarized.

As shown in Figure 11, the groups of users with the lowest number of errors are in the age range between 31 and 35 years. Also, this group presents a low median absolute deviation. This result can be related to the knowledge of similar programming tools.

Table 5. Median errors (amount) for males and females.

Value	Male	Female
Median Error Rotating	0.1	0.1
Median Error Grabbing	0.5	1.0
Median Error Placing	0.2	0.0

Figure 11. The median amount of user errors according to the age range (with median absolute deviation). The blue bars represent the median amount of user errors, given different groups of users (according to their ages).

In Figure 12, the median user errors for each gender are presented.

As in Figure 12, the female users with the lowest number of average errors are in the range of 30 to 40 years old with a relatively low standard deviation, and the opposite happens in the range between 40 and

Figure 12. Median user errors according to the age range by each gender (with median absolute deviation). The blue bars represent the number of user errors for male subjects, and the purple bars represent the number of user errors for female subjects, given different groups of users (according to their ages).

50 years old. The figure also shows the male users over 40 years old are the subjects with fewer errors on average. The previous result can be related to a more accurate sequence of gestures' performance, which is probably because these users were faster and precise to perform the gestures.

Finally, in Figure 13, a comparison considering the number of errors is shown to evaluate the influence of user errors on the overall performance.

Figure 13 shows that those users with 0 and 2 errors have the best average time performance. Also, the time to perform the task was longer for the users with more than 3 errors (where the maximum amount of errors was four). This extra time indicates a correlation between user errors and time to perform the task.

The statistical evaluation, in this case, was performed against the mean value of user errors, 1.1. The values obtained were Wilcoxon Statistic = 137, $p > 0.01$ (two-tailed), indicating that there is no statistical significantly evidence to affirm the number of user errors are more than the average value of the time performance for our approach.

System Errors:

The final metric to be analyzed corresponds to the errors generated by the system. These errors are associated with failure in identifying a performed

Figure 13. Median time to perform the task according to the number of user errors (with median absolute deviation). The purple bars represent the time to perform the task, given different amounts of user errors.

gesture correctly. The results for this metric are shown in Figure 14.

Figure 14 presents the influence of the system errors in the overall required time to complete the tasks. As it can be seen, the number of system errors in the range of 7 to 10 made the task more difficult, but with a high standard deviation, which indicates the average time performance, in this case, is not directly influenced by system errors. The error range between 4 and 6 has the best time performance. This range indicates that the amount of system errors is not directly related to the time to perform the task, indicating that the users can quickly overcome these problems.

The statistical evaluation, in this case, was performed against the mean value of system errors, 7. The values obtained were Wilcoxon Statistic = 1802, $p > 0.01$ (two-tailed), indicating that there is no

statistically significant evidence to affirm that the number of system errors is more than the average value of the time performance of our approach.

Comparison of qualitative and quantitative results

The qualitative results show that most users consider the presented approach more intuitive and user-friendly than the traditional text code-based interaction to create multi-threaded software, especially in terms of learning time and interaction, as can be seen in Figure 8 and Figure 9.

In the case of the qualitative evaluation, it can be observed that the users with the lower number of errors are, in general terms, the users that performed the task faster. This difference can be seen in Figure 10 and Figure 12, where the users between 26 and 35 years present the average best results.

It can be seen that the users with poorest performance (errors and time) are the ones that evaluated the interface with the lowest marks comparing the quantitative and qualitative results (using as reference the four figures discussed above), which corresponds to the users in the age range between 41 and 45 years old.

Table 6 shows the values obtained for our sets of experiments.

In the next section, the conclusions are presented.

CONCLUSIONS

This paper presented a novel 3D hand gesture-based programming environment was presented, designed mainly for multi-thread applications. The complete definition of the interface involved graphic elements and advanced interaction techniques. The supported gestures and how to interact with the interface were

Figure 14. Median time according to the amount of system errors clusters (with median absolute deviation). The blue bars represent time to perform the task, given different amounts of system errors.

Table 6. Statistical significance values for the experimental data obtained.

Experimental Data	Amount of Subjects Involved (N)	Mean response value	Wilcoxon Statistical	P-Value
Questionnaire Section 1	29	3.5	6	0.00004
Questionnaire Section 2	29	3.5	14	0.00011
Average tasks' performance time	29	93.1 (seconds)	134	0.1161
Average amount of errors	29	1.1	137	0.13286
Amount of system errors	29	7	180	0.83818

analyzed, and a prototype graphical user interface. This model was presented but not evaluated on [27].

Also, an evaluation experiment was described, analyzing the evaluation procedure and metrics. The qualitative evaluation was based on a questionnaire to retrieve the users' opinions regarding the interface. The quantitative evaluation was based on three parameters: Time to perform a specific task, user errors, and system errors during the interaction.

The questionnaires analysis revealed that the users gave the system a positive evaluation, particularly, the system's performance and how intuitive it was, confirming that a gesture-based interface is more comfortable and easy to understand by the users.

According to the results obtained, it can be said that users of both genders have relatively similar performance on the performed execution times and amount of user errors. Also, the system errors do not significantly affect the overall task execution time.

The results presented confirm that a gesture-based interface plus a 3D graphical development environment provide significant advantages in terms of performance and, consequently, can improve the user experience and the learning time of new programming techniques. Also, the 3D visualization provides a better understanding of the task of designing multi-thread software.

REFERENCES

- [1] K.P. Fishkin. "A taxonomy for and analysis of tangible interfaces". *Personal and Ubiquitous Computing*. Vol. 8 N° 5, pp. 347-358. 2004.
- [2] E. Guettard, S. Jandziak, C. Lombard, N. Podevin, V. Varane and P. Dumont. "Virtual reality from the keyboard/mouse couple to Kinect". *Annals of Physical and Rehabilitation Medicine*. Vol. 54 N° 10 suppl 1, pp. 239. 2011.
- [3] M. Chau, A.Wong, M. Wang, S. Lai, K.W. Chan, T.M. Li and W.K. Sung. "Using 3D virtual environments to facilitate students in constructivist learning". *Decision Support Systems*. Vol. 56, pp. 115-121. 2013.
- [4] R. Wagner, S.D. Piovesan, P. Passerino, L. Maria and J.V. de Lima. "Using 3D virtual learning environments in new perspective of education". *International Conference on Information Technology Based Higher Education and Training (ITHET)*, pp. 1-6. 2013.
- [5] P.H. Wang, J.D., Collins, G.N. China, H. Jiang, X. Tian, M. Girkar and H. Wang. "EXOCHI: Architecture and programming environment for a heterogeneous multi-core multithreaded system". In *ACM SIGPLAN Notices*. Vol. 42 N° 6, pp. 156-166. 2007.
- [6] S. Ryoo, C.I. Rodrigues, S.S. Bagsorkhi, S.S. Stone, D.B. Kirk and W.M.W. Hwu. "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA". *13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pp. 73-82. 2008.
- [7] J. Nickolls, I. Buck, M. Garland and K. Skadron. "Scalable parallel programming with CUDA". *Queue*. Vol. 6 N° 2, pp. 40-53. 2008.
- [8] R. Dolbeau, S. Bihan and F. Bodin. "HMPP: A hybrid multi-core parallel programming environment". *Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2007)*. 2007.
- [9] N. Kumar and I. Benbasat. "The effect of relationship encoding, task type, and complexity on information representation: An empirical evaluation of 2D and 3D line graphs". *MIS Quarterly*, pp. 255-281. 2004.
- [10] C. Ratti, Y. Wang, H. Ishii, B. Piper and D. Frenchman. "Tangible User Interfaces (TUIs): A novel paradigm for GIS". *Transactions in GIS*. Vol. 8 N° 4, pp. 407-421. 2004.
- [11] C. Ratti, Y. Wang, H. Ishii, B. Piper and D. Frenchman. "Tangible User Interfaces (TUIs): A novel paradigm for GIS". *Transactions in GIS*. Vol. 8 N° 4, pp. 407-421. 2004.
- [12] G. Vazquez, A. Diaz Pace and M. Campo. "Reusing design experiences to materialize software architectures into object-oriented designs". *Information Sciences*. 2010.
- [13] A. Marcus, L. Feng, and J.I. Maletic. "3D representations for software visualization". *2003 ACM symposium on Software visualization*, pp. 27-ff. DOI: 10.1145/774833.774837
- [14] C. Geiger, W. Mueller and W. Rosenbach. "SAM-an animated 3D programming language". *Symposium on Visual Languages, Proceedings*, pp. 228-235. 1998.

- [15] T. Okamura, B. Shizuki and J. Tanaka. "Execution visualization and debugging in three-dimensional visual programming". 8th International Conference on Information Visualisation, Proceedings, pp. 167-172. 2004.
- [16] G. Ren and E. O'Neill. "3D marking menu selection with freehand gestures". IEEE Symposium on 3D User Interfaces (3DUI), pp. 61-68. 2012.
- [17] T.M. Takala, P. Rauhamaa and T. Takala. "Survey of 3DUI applications and development challenges". IEEE Symposium on 3D User Interfaces (3DUI), pp. 89-96. 2012.
- [18] M.D. Linderman, J.D. Collins, H. Wang and T.H. Meng. "Merge: A programming model for heterogeneous multi-core systems". ACM SIGOPS Operating Systems Review. Vol. 42 Nº 2, pp. 287-296. 2008.
- [19] J.J. Harrow. "Runtime checking of multithreaded applications with visual threads". In SPIN Model Checking and Software Verification. Springer Berlin Heidelberg, pp. 331-342. 2000.
- [20] M.B. MacLaurin. "The design of Kodu: A tiny visual programming language for children on the Xbox 360". ACM Sigplan Notices. Vol. 46 Nº 1, pp. 241-246. 2011.
- [21] B. Karpak, R.R. Kasuganti and E. Kumcu. "Multi-objective decision-making in supplier selection: An application of visual interactive goal programming". Journal of Applied Business Research (JABR). Vol. 15 Nº 2, pp. 57-72. 2011.
- [22] G. Biggs and B. MacDonald. "A survey of robot programming systems". Australasian Conference on Robotics and Automation, pp. 1-3. 2003.
- [23] I.A. Baky. "Solving multi-level multi-objective linear programming problems through fuzzy goal programming approach". Applied Mathematical Modelling. Vol. 34 Nº 9, pp. 2377-2387. 2010.
- [24] J. Hurtienne, C. Stobel, C. Sturm, A. Maus, M. Rötting, P. Langdon and J. Clarkson. "Physical gestures for abstract concepts: Inclusive design with primary metaphors". Interacting with Computers. Vol. 22 Nº 6, pp. 475-484. 2010.
- [25] J.R. Lewis. "IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use". International Journal of Human-Computer Interaction. Vol. 7 Nº 1, pp. 57-78. 1995.
- [26] B. Rosner, R.J. Glynn and M.L.T. Lee. "The Wilcoxon signed rank test for paired comparisons of clustered data". Biometrics. Vol. 62 Nº 1, pp. 185-192. 2006.
- [27] R.A. Herrera-Acuña, V. Argyriou and S.A. Velastin. "Toward a 3D hand gesture multi-threaded programming environment". International Visual Informatics Conference. Springer, Cham. Vol. 8237, pp. 1-12. 2013.