A Mixed Reality Game using 3Pi Robots - "PiTanks"

Hugo Costa, Peter Cebola, Tiago Cunha
Department of Electrical and Computer Engineering,
Faculty of Engineering, University
of Porto
Porto, Portugal
{ee10112, ee10074, ee10203}@fe.up.pt

Armando Sousa
INESC TEC (formerly INESC Porto)
and Faculty of Engineering, University
of Porto
Porto, Portugal

***Abstract*—In the growing field of Robotics, one of the many possible paths to explore is the social aspect that it can influence upon the present society. The combination of the goal-oriented development of robots with the interactivity used in games while employing mixed reality is a promising route to take in regard to designing user-friendly robots and improving problem solving featured in artificial intelligence software. In this paper, we present a competitive team-based game using Pololu's 3Pi robots moving in a projected map, capable of human interaction via game controllers. The game engine was developed utilizing the framework Qt Creator with C++ and OpenCV for the image processing tasks. The technical framework uses the ROS framework for communications that may be, in the future, used to connect different modules. Various parameters of the implementation are tested, such as position tracking errors.**

Keywords-3Pi Robot, Mixed Reality, Game Engine, Qt, Robotics, Robotic Demonstrator, ROS

I. INTRODUCTION

The technology of mixed reality is not a novelty in the field of Robotics, but it is a subject with promising development. The definition of mixed reality is often ambiguous, merging with the one of augmented virtuality and augmented reality. Using the definition in [1], mixed reality is the usage of computer generated elements on the real world or it can be the opposite; representations of the real world to enhance a simulation, e.g., a photo on a 3D graphic model. The first example before is more representative of the definition of augmented reality by the same author, while the latter is of augmented virtuality. As such, our project resembles more closely mixed reality, taking in account that a program-generated image is projected to show various objects necessary to the game. Using such approach enables the use of features like shooting without implementing physical systems in the robot, sparing physical resources.

The development of any game concept follows the division defined in [2], where the two essential groups of elements are: rules and goals, which defines the founding guidelines of interaction of the different elements of the game and the objectives to accomplish in order to win and props & tools, which complements the rules, assisting them to make a functional game. PiTanks uses all those basic elements, having a great focus on multiplayer.

II. RELATED WORK

Projects with the same basis of concept were created before. Anki Drive of Anki Inc. [3], now acquired by Apple, is a racing game that utilizes an embedded system on miniature motorized cars using processing power from an iPad or iPhone. The small cars run on a track printed in a prepared mat and are controlled in a simulation that runs on the iPad/iPhone. Different weapons are available for the various existent models (but only on the simulated environment) and the racers can be either human or AI controlled, up to 4 cars at the same time.

Another project, developed by the Play Research Studio Interactive Institute, is the "Pirates!" game [4], which merges electronic hardware and elements of reality. It consists of a multiplayer game where players take the role of pirate captains with the objective of accumulating experience points and gold by completing missions and fighting other players in naval battles. The game arena is defined by a chosen region in the real world, having locations with radio frequency transmitters as beacons to map the virtual islands. The players use laptop computers with a graphical interface as the game controllers, simulating the boat movement with walking in real life.

Also using robots in a projected environment are the Robot ARena from Polytechnic School of the University of São Paulo in Brazil and the Augmented Coliseum from the University of Electro-Communications of Japan. The Robot ARena [5] is a hardware and software infrastructure with a similar concept of the proposed one (referring to the FootBot Arena prototype presented in that paper): remote controlled robots moving in a projected environment, enabling interaction between the virtual and real world. The robots used are Mindstorms NXT from Lego, having Lego pieces on top to serve as markers for localization. The projection is made on an acrylic table with a camera on top, serving as the hardware for the localization and interactivity with the virtual elements. The game engine is based on the open-source engine, created in Java.

The Augmented Coliseum [6] is also a combat game involving small robots with projected features in order to create augmented reality. The system uses two models of the robots (real and virtual) to coordinate the robot movements. Tracking is done using brightness sensors on the top of the robots in conjunction with a projected fiducial marker in order to obtain position and orientation of the robots. With those data, the model of the real object is updated and if the robot is moved in the simulation, the real robot is moved accordingly. The objective of the game is to destroy the other robots utilizing weapons like lasers or missiles, also having a shield to defend from incoming fire. All these functionalities are represented with the projection of generated images, giving information and feedback to the users.

III. System Description

PiTanks is a multiplayer game that utilizes mixed reality with the use of robots and projected images. Players interact with the game utilizing the game controllers, battling between themselves or in teams. Resorting to a camera, snapshots are taken from the game area (which is projected on the ground), informing the main program about the location of the physical robots and permitting the update of the game state by changing the projected scene. Various configurations are possible, such as the numbers of possible players (up to eight robots in the same map), the choice between two games modes (Timed Match and Last Team Standing) and the decision between premade maps or the creation of a new one via an application in the software. The photo-based tracking system is a dedicated program using created markers on the top of the robots.

The development of the project involved diverse technologies from both the software front and the hardware side. Various possibilities were studied, having the defined choices explained and shown in the next sections.

IV. System Breakdown

The system breakdown is shown in Fig. 1 and the software architecture in Fig. 2. Game controllers are used by the players to control the robots, having their inputs transmitted to the game engine via ROS. The game engine is informed of the position of the robots analyzing the images received from the camera. It conjugates then those data with the inputs from the controllers to process the game state (see if a collision occurred, calculate bullet paths, wall destruction, etc.) and projects the output to the playing field.

A. Hardware

1) Robot – Polulu 3Pi [7]: The robot used is the 3Pi, developed by Pololu. It's a 9.4 cm diameter wide (hence the name) and two-wheeled robot with an ATmega328P microcontroller with 32kB flash memory, 2kB RAM and 1kB EEPROM. The wheels are locomoted by two micro metal gear motors (with a plastic ball caster on the front of the robot as a free wheel). It also includes five IR reflectance sensors, three push buttons, a buzzer, one green and one red LED, and 8x2 LCD screen. All of this is supplied by four AAA batteries. This robot was chosen for being low-cost and capable of differential traction (the closest to tracks), enabling rotation on itself. In order to enable wireless communication using XBee and implementing the marker for localization, the robot was modified, placing a board for the module circuitry and to support the marker, as shown in Fig. 3.

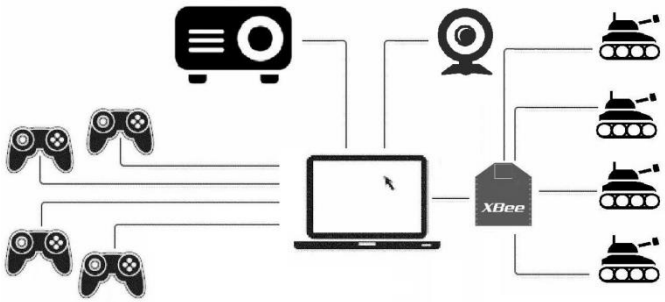

Figure 1. System Breakdown Structure.

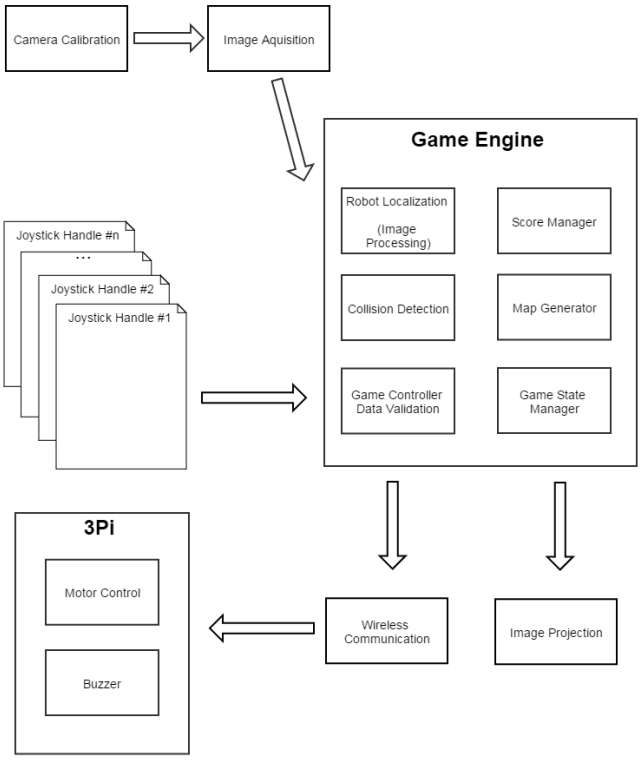

Figure 2. Diagram of the software architecture.

2) Camera – PlayStation 3 Eye [8]: To locate the robots in the projected map, the camera used is PlayStation Eye for the PlayStation 3 system. It possesses a field view of 56 degrees that can go up to 75 with the incorporated zoom lens and a frame rate of 60 frames per second with a video capture of 640x480 pixels. It also provides uncompressed video format, useful for more precise and efficient image processing.

3) Projector – Hitachi ED-X3270A [9]: A Hitachi ED-X3270A is used for the projection of the virtual elements. It's capable of a resolution of 1024x768 pixels and a focus distance between 0.9 and 11.0 meters. The maximum display size is 7.62 meters, which gives enough space for the robots to move on the projection of the map.

4) Controller - Gamepad controllers [10, 11]: Various game controllers are used, such as Trust GXT 24 Compact Gamepad, Logitech RumblePad2, and NGS Maverick. Having two analog sticks, one directional pad and at least 8 different push buttons (plus two from the analogs), they give all the necessary inputs to the users to control the robots.

5) Communication – Xbee [12]: The wireless communication between the game engine and the robots is made with the XBee 802.15.4 radio module from Digi International, having a range of 30 meters and a wireless data rate of 250 kilobits per second. Because communication time is a constraint in the project development, radio frequency proves to be a good solution, securing a quick and reliable way to transfer data, while permitting multipoint communication.

B. Software

1) Coding - C++: As a result of the real-time constraints in PiTanks, we needed to choose a language that could meet those restrictions. Many engines and games are written in C++ and it offers better performance than Java on this topic.

2) Image Processing – OpenCV [13]: For the necessary image processing to identify the robots, OpenCV proved to be a good solution. Having an extensive library of ready-made funcions either in C, C++ or Python, it supplied all the tools needed for this task. It's opensource and compatible with Windows, Linux and Android.

3) User Interface – Qt [14]: Due to the project timelines and time-constraints, building our own engine from the ground up was out of the question. A study and discussion of the available programs for designing an engine was made, and the team settled for Qt because of its easy GUI programming (Qt Designer), useful container classes that implemented everything needed, extensive documentation with examples and the fact that it's all embedded in the same IDE.

4) Robot Operating System (ROS) [15]: In this project, the team implemented a communication network based on ROS. This was due to three main things:

- ROS makes the communication setup abstract and modular (easy to build on).
- ROS has pre-defined message types which are useful for transmitting speeds to the 3Pi robots. (Debug is also easy to do and visualization is intuitive.)
- The subscriber/publisher model ROS is built on is abstracted to where the data is coming from and how. It is known that there is a specific type of data on each topic, so it can easily be listened to it.

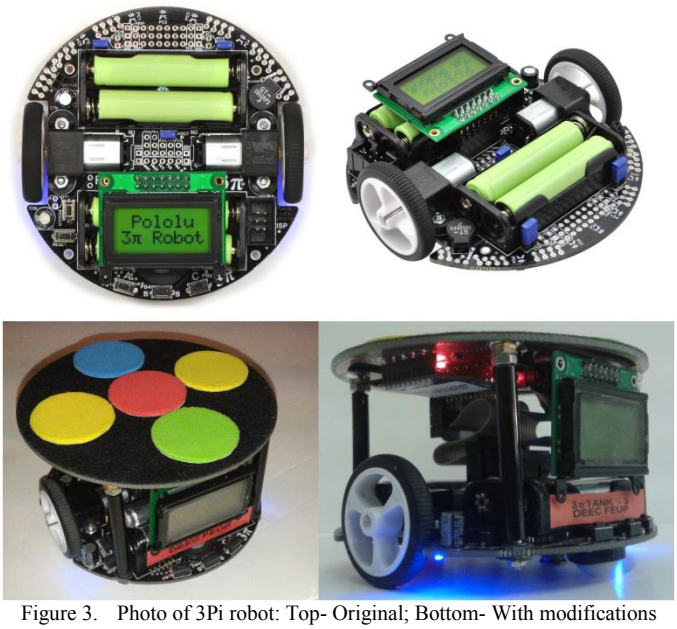

Figure 3. Photo of 3Pi robot: Top- Original; Bottom- With modifications

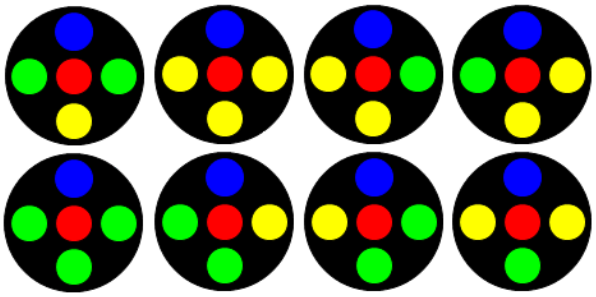

Figure 4. Color markers used for robot identification

V. Software Development

A. Image Processing

The detection of the 3Pi robots is achieved using the localization markers located above them. They are constituted with five colored circles in a cross pattern on top of a larger black base (as seen on Fig. 4). The blue circle always represents the front of the robot, the red signals the center of the robot and the combination of the five identifies which robot is being seen. This kind of pattern was taken from [16]. The position on the map is calculated using an average of the centers of the clusters of the colored circles and the orientation is given with the average of the angles of each circle to the center corrected to be placed in the first quadrant. For the color segmentation, a look-up table (LUT) is implemented to store the information about the colors, also permitting to select the color by clicking on the correct place in the image. The colors are detected resorting to RGB images, using only the five most significant bits, enabling the storage of the LUT in cache memory, improving CPU time and thus having better performance.

B. Game Setup

Upon the start of the program, the control window appears (from here on it will be called control), as seen on Fig. 5. From here, every functionality of the game can be chosen, such as the number of players, the map, the game mode, and the calibration parameters. To start a game, it must be selected a valid configuration (all the robots that will participate in the game must be seen by the camera in the defined game area) and click on the Start button. Then, depending on the game mode selected, the game engine will behave differently.

C. Game Agents

The game consists of three main agents: the robots, which act as tanks, the bullets, that can be fired, and the walls to create interactivity with the map. Each of the agents is described below:

1) PiTanks: The PiTanks robots are the characters in the game, controlled by game controllers. The tanks navigate the field at approximately 0.10 m/s to a configurable max speed currently of 0.30 m/s, damageable walls and trying to defeat opposing tanks. Each tank has its own life, and when that life reaches zero the robot spins around itself for a while and emits a sound, while graphically a red ring around the robot is projected to signal the user the tank has lost all life. If Timed Match mode is selected, instead of permanently being

destroyed, a cooldown timer is used, where the robot can't attack or be attacked, with only being able to move around the map. When the counter stops, it is restored with the same amount of life at the beginning. A turbo button was added to allow the users to navigate the field faster. When this button is pressed the shoot button is disabled but the robot moves almost twice as fast as it normally would. This function was implemented to allow players to evade incoming enemy bullets.

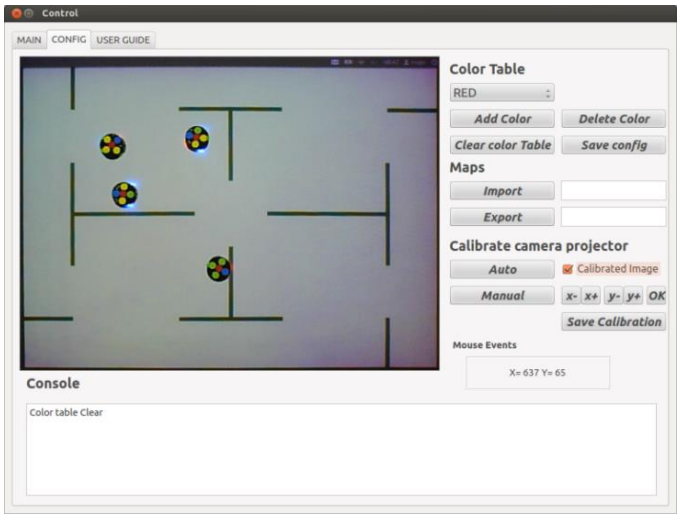

Figure 5. Screenshot of the control program's window.

2) Bullets: Bullets are also objects in the game, represented by red ellipsis on the map projection. They are shot when the user presses the corresponding button on the joystick, having a maximum frequency of one bullet per second. When the bullet hits an object on the map other than bullets it disappears and is deleted from the object list. An animation of an explosion is implemented to graphically warn the players of a colliding bullet.

3) Walls: The final elements of the game engine are the walls, which can't be crossed by the tanks. They can be either destructible, with an associated counter for the durability of the wall that decreases when it collides with a bullet, or non-destructible. The wall's durability change is visible through a gradient of colors depending on the current percentage, starting with black and becoming redder when damaged.

D. Game Engine

A game engine is framework which facilitates all operations in the game, like loading maps, control in-game states, manipulate the environment, generate results, etc. A description of chosen approaches is detailed hereafter.

1) Scores: Individual scores are kept for each robot. The highest scoring player will be the one with fewer hits and with better accuracy (ratio between shots fired/shots taken). For eventual draws, shot accuracy is used to resolve them.

2) Collisions: A collision detection system was implemented by the group based on the one supplied by Qt's framework in conjuction with odometry and a basic position predictor. This was performed to help with the lag between what is happening in reality and what is being simulated, like, for example, a robot bumping against a faraway wall.

E. Game Window

The game GUI has two windows: one that the user controls (as mentioned in section V-B) and another that holds the image to be projected. On the window that the user controls, the user can select one of the three tabs, either for configuration, a guide for new users or the main window. Each tab has different functions: the main tab allows the user to select various parameters already mentioned, but it also allows for data loading and export, also having a console to show relevant messages; the configuration tab allows map handling, such as importing and exporting, camera calibration and it also allows the user to add colors to the LUT.

F. Utilities

1) Map Generator: A tool developed by the team that allows the user to draw its own maps. Made entirely with Qt, it allows drawing horizontal or vertical lines in a map or make changes to an existing one. This is achieved by inserting the coordinates referring to the start and end point of the line and the durability of a wall in the program. When a map configuration is saved, it is exported on a text file that contains those data which can be imported later.

2) Joystick Handler: A program that captured the joysticks in the USB ports was created by the authors. The capture program worked in a multi-threaded environment and communicated over ROS to the game engine.

G. Communications

1) IEE 802.15.4 Communication: In order to send the information of the joysticks to the robots, a package with the data formatted for the 3Pi to read (in API2 mode) is transmitted. All robots have an different ID, distinguishing the messages for the various robots. These packages are sent constantly to ensure the continuous correct movement of the robots.

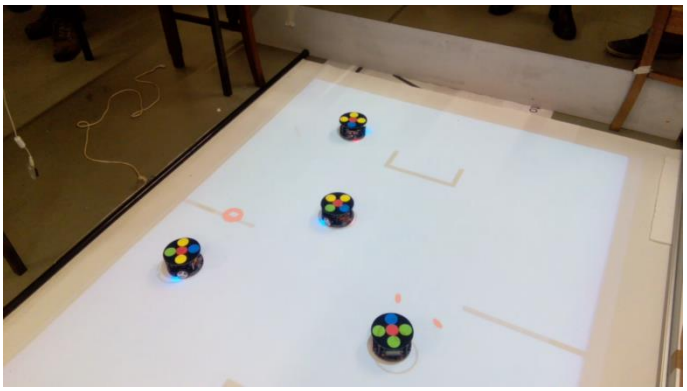

Figure 6. Photo of a PiTanks game session.

2) ROS Nodes: The implemented ROS architecture consists of three nodes: one that publishes interpreted commands of the joysticks into different topics according to the number of joysticks currently present and connected, one

that listens to the commands, runs them through the engine and then publishes on another topic, and one that deals with the XBee communication, listening to the previous topic and sending data to all the 3Pi. For the joystick node, three types of joysticks where used. While ROS provides a package for interpreting commands from the last two controllers, it did not interpret the generic game controller from Trust, so the team had to implement their own joystick manager package (that would read information from all three types of joysticks and publish them). The information that was published was a geometry message that included the velocity to give the 3Pi (linear x, linear y and angular z). To reuse this type of message and to simplify the code, the linear z was used to indicate that either the shoot button or the turbo button was being pressed.

VI. Tests and Analysis of Results

A test regarding the precision of the localization of the 3Pi robots was executed. Projecting a 640x480 image of a seven by nine grid on the game area, a robot is moved into the line's intersections, measuring a total of 63 points in each experiment. The results of the position error in each position of the test image (all relative to the center of the robot) is shown in Fig. 7 and Fig. 8 and the orientation error is shown in Fig. 9 and Fig. 10. Analyzing Fig. 7 and 8, we can observe that the error is inferior on the top center of the test image. This fact can be explained with the distortion of the camera lens, having greater error when the robot is farther from the center of the lens. In the case of Fig. 9 and 10, this effect doesn't influence the angle measurement, having less error in the borders.

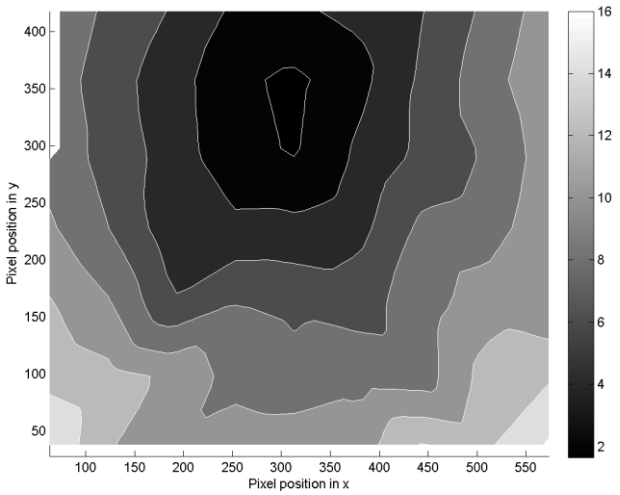

Figure 7. Position error in each pair of coordinate x,y of the test image in pixels for a position angle of the robot of zero degrees (value mapped in grayscale bar).

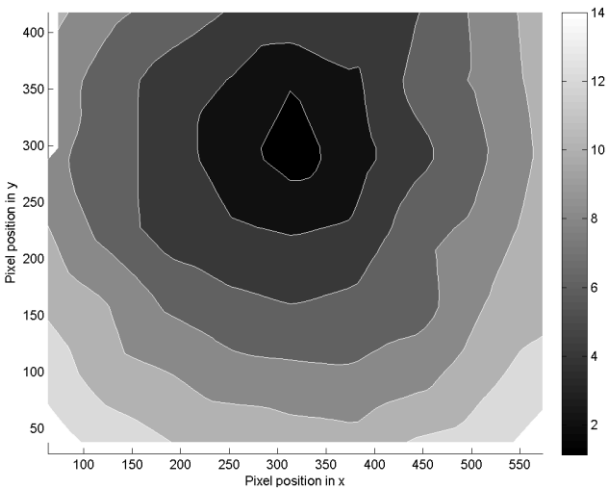

Figure 8. Position error in each pair of coordinate x,y of the test image in pixels for a position angle of the robot of 90 degrees (value mapped in grayscale bar).

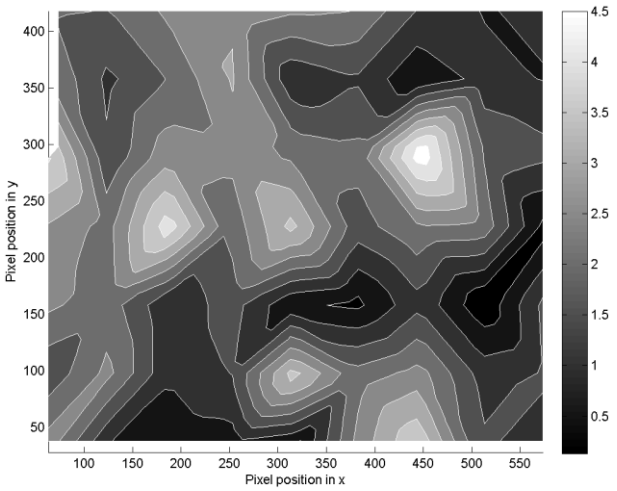

Figure 9. Orientation error in each pair of coordinate x,y of the test image in degrees for a position angle of the robot of zero degrees (value mapped in grayscale bar).

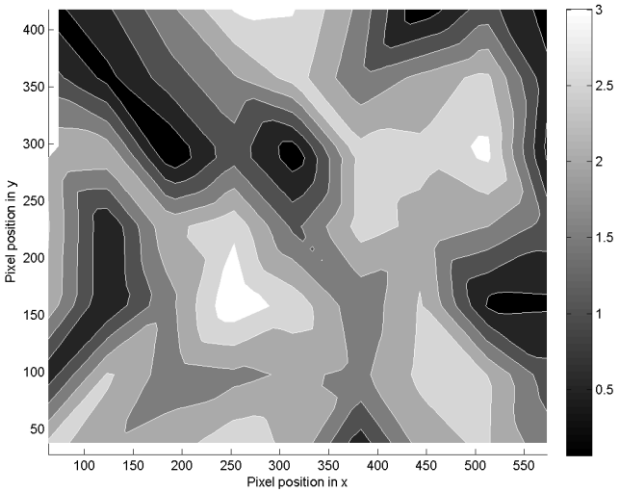

Figure 10. Orientation error in each pair of coordinate x,y of the test image in degrees for a position angle of the robot of 90 degrees (value mapped in grayscale bar).

VII. SURVEY

To see the impact caused by our robotic demonstrator and how easy is to understand the rules of the game and underlying concepts, a survey was conducted, with the results shown on Fig. 11. The sample gathered was composed of 27 seventeen-year-old high-school students. The survey was composed by five statements (as written in Fig. 11), where the respondents would rate their belief on a scale of options, with 5 being completely agreeing to the statement and 1 being completely disagreeing. Analyzing Fig. 11, one can take the illation that PiTanks creates an awareness and interest in Robotics on people that don't have Electrical Engineering as a career choice: 86% had chosen no interest or dismissing the possibility of choosing Electronic Engineering, but most of those learned from the demonstration and liked the game. The concept and controls of the game were also considered to be easy to grasp, having also mentioned that some key aspects and technologies of robotics surrounding the game were learned.

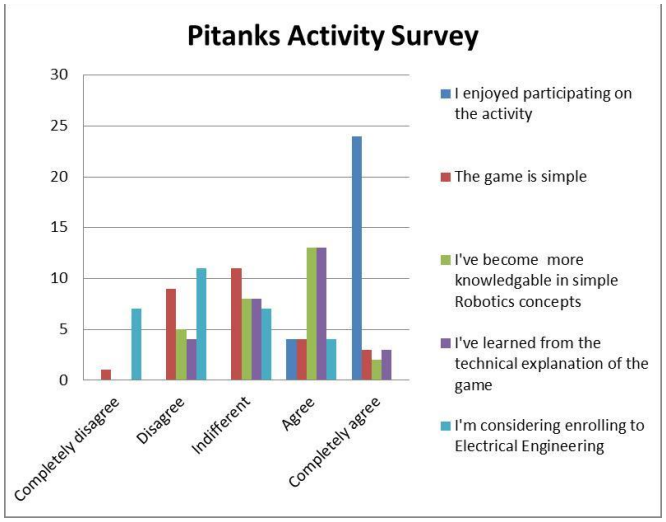

Figure 11. Result of a survey to inquire the opinions of involved users.

VIII. CONCLUSIONS

The purpose of this work was to create a low-cost platform to run a game for a younger audience. With all the material presented above, we presented a simple, cheap and fun game engine with simple mechanics that take advantage of the 3Pi robots that possess responsive commands with simple and straight-forward mechanics, since that is imperative for interaction with younger audiences. Not only is it effective as a robotics demonstrator, but as it was also built in ROS, it allows the ability to be built on and be interfaced with Artificial Intelligence, allowing experiments with Swarm Robotics, or the implementation of other modules on top of the existing ones. As such, this project makes the STEM areas (Science, Technology, Education and Mathematics) more attractive to the general public, making them curious for the introduced technologies, which is the philosophy applied to this work in order to involve others into the field of Robotics.

ACKNOWLEDGMENT

This work is partially financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «FCOMP-01-0124-FEDER-037281». The authors also wish to thank the Dept. of Electrical and Computer Engineering of FEUP for a part of the financing of this work.

REFERENCES

[1] P. Milgram and H. Colquhoun Jr., "A taxonomy of real and virtual world display integration," *Mixed reality: Merging real and virtual worlds,* pp. 5-30, 1999.

[2] J. P. Zagal, M. Nussbaum, and R. Rosas, "A model to support the design of multiplayer games," *Presence: Teleoperators and Virtual Environments,* vol. 9, pp. 448-462, 2000.

[3] B. Sofman, H. Tappeiner, M. M. Palatucci, and P. L. DeNeale, "Integration of a robotic system with one or more mobile computing devices," ed: Google Patents, 2013.

[4] S. Björk, J. Falk, R. Hansson, and P. Ljungstrand, "Pirates!–using the physical world as a game board," in *Proceedings of interact*, 2001, pp. 423-430.

[5] D. Calife, J. L. Bernardes Jr, and R. Tori, "Robot Arena: An augmented reality platform for game development," *Computers in Entertainment (CIE),* vol. 7, p. 11, 2009.

[6] M. Kojima, M. Sugimoto, A. Nakamura, M. Tomita, M. Inami, and H. Nii, "Augmented Coliseum: An Augmented Game Environment with Small Vehicles," in *Tabletop*, 2006, pp. 3-8.

[7] Pololu. *Pololu 3pi Robot*. (Accessed December 2014). Available: https://www.pololu.com/product/975

[8] S. C. E. A. LLC. *PlayStation®Eye Camera*. (Accessed December 2014). Available: http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html

[9] Hitachi, *Liquid Crystal Projector ED-S3170A/ED-X3270A USER'S MANUAL* vol. 1 (Basic).

[10] T. D. L. Accessories, "GXT 24 Compact Gamepad," ed, 2014.

[11] Logitech. *Rumblepad 2 - Logitech Support*. (Accessed December 2014). Available: http://support.logitech.com/product/rumblepad-2

[12] D. I. Inc., "XBee® Multipoint RF Modules - Embedded RF Modules for OEMs," ed.

[13] Itseez. (Accessed December 2014). *OpenCV*. Available: http://opencv.org/

[14] D. plc. *Qt Project*. (Accessed December 2014). Available: http://qt-project.org/

[15] O. S. R. Foundation. *ROS.org | About ROS*. (Accessed December 2014). Available: http://www.ros.org/about-ros/

[16] P. Costa, A. P. Moreira, A. Sousa, P. Costa, S. Gaio, P. Marques, *et al.*, "5dpo Team Description for RoboCup 2006."