Robot Devastation: Using DIY Low-Cost Platforms for Multiplayer Interaction in an Augmented Reality Game

David Estevez, Juan G. Victores, Santiago Morante, Carlos Balaguer
Robotics Lab research Group
Universidad Carlos III de Madrid
Email: {destevez, jgcvicto, smorante, balaguer}@ing.uc3m.es

***Abstract*—We present Robot Devastation, a multiplayer augmented reality game using low-cost robots. Players can assemble their low-cost robotic platforms and connect them to the central server, commanding them through their home PCs. Several low-cost platforms were developed and tested inside the game.**

***Keywords*—augmented reality, diy, low-cost, multiplayer interaction, robots, game**

I. Introduction

An increasing trend in educational and hobby robotics has been seen in the past years. Kits for building Do-It Yourself (DIY) robots are now common, and low-cost electronic boards have spread in last years. These educational or hobbyist kits usually leverage from open source licenses, in order to ease the use and development of these products, and also increase the interest of people in electronics, mechanics and programming. Within this context, Robot Devastation is an open source game where DIY robotic developments are integrated as real-world 'avatars' of the players. The idea is to integrate real user-made robots with this Augmented Reality (AR) game, through the use of fiducial elements (e.g. AR markers). These robotic platforms can interact in a common virtual space and be controlled from off-the-self devices (a mockup is shown in Fig. 1).

Azuma et al. presented the most common applications of existing AR systems in [1]. Among them, we can find medical applications, including training and aid for surgery or applications in manufacturing and repairing such as live view of 3D schematics and instructions on top of the actual objects. The last application described in the aforehead mentioned work is entertainment and gaming, which is the topic this paper is focused in.

Later, in [2], Azuma et al. updated this list of applications of AR introducing three new applications: outdoor and mobile AR systems; collaborative AR, very useful in environments such as meetings or collaborative product design and commercial applications, like sports broadcasting or advertising.

In this work we present Robot Devastation (Fig. 4), a robot combat game that uses teleoperated robots as avatars for the players. This aspect distinguishes it from most of the AR games, which are purely virtual, meaning that all entities controlled by the player do not correspond to any real world object or are passive elements such as markers or instruments (e.g. a "magic wand"). This way all the dangerous or expensive elements of the robot combat can be virtual, and added to the real world using the augmentation capabilities of AR. The architecture of our game allows seamless integration of low-cost robotic platforms in this multiplayer First-Person Shooter (FPS). Interaction occurs within an Augmented Reality environment where gaming elements such as weapons and ammunition, as well as menus and guidelines, are superposed on the stream of images that arrive with the robot's perspective.

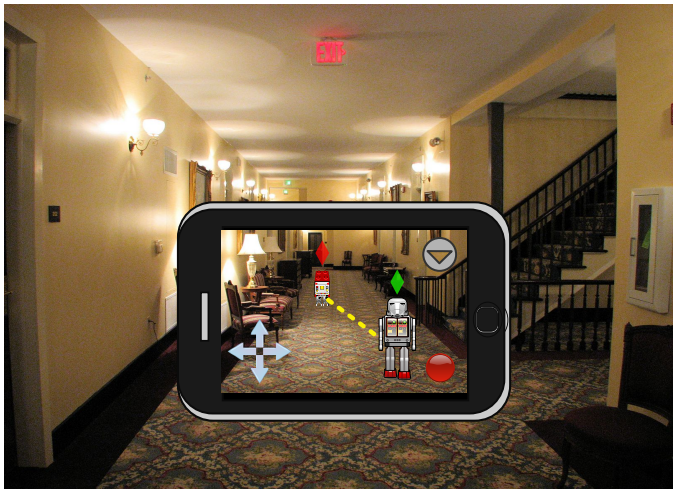

Fig. 1. Robot Devastation gameplay mockup from smartphone. The robots seen on the screen represent the players' own robots.

II. State Of The Art

One of the widespread applications of AR are games and entertainment systems. For the purpose of studying them in this work, they are classified as research AR games and commercial AR games. Research AR have as their main goal researching the possible applications of AR concepts to gaming. On the other hand, the main objective of commercial AR games is to be released to the general public to obtain an economic profit. The later group is becoming very popular lately, as the cost of the hardware required to implement them is decreasing, but few information or technical data is often known about them.

A. Research Games

Research games are games made as a proof of concept for an AR development, usually with no intention of reaching an end user. One of the first examples of these kind of games was developed by Ohshima et al., called AR2 Hockey [3]. AR2 Hockey is a collaborative game were two players play a real-time, interactive game of air hockey. The puck is virtual, and the players can see it through their Head-Mounted Displays (HMD), and hit it using either real or virtual strikers.

AR Quake [4] is an adaptation of the classic Quake game. In this version, the player is equipped with a HMD and a prop gun with haptic feedback simulating the gun recoil, and he is allowed to move freely outdoors. The game character is controlled through his movements in the real world, which has been previously mapped and used as a base to generate the game virtual environment.

Monkey Bridge [5] demonstrates how autonomous agents can be integrated in AR games. In this indoor, two-player game, two autonomous virtual characters have to cross a bridge over virtual water. Each player, in turns, adds new sections of the bridge using fiducial markers, helping those virtual characters to cross to the other side of the table, without falling into the virtual water.

Augmented Reality has been also applied to already existing table games such as Monopoly [6], to detect the game board and tiles, as well as the player's pawns. Their system follows the game flow, and offers both visual enhancements and a tangible interface to interact with the game by means of the different pawns.

Knoerlein et al. developed an AR ping pong game for two players [7]. The virtual ball was presented on top of the real table by means of a HMD, and the users received haptic feedback using a modified table tennis bat whenever they hit the ball.

Oda et al. presented in [8] a multiplayer AR racing game. One player had the role of the driver, and the other players could interact with the racing car by adding waypoints and obstacles to the virtual track, which was overlaid on top of fiducial markers. Therefore, several cameras captured the real world, and several displays showed the augmented world to the players. The racing car was controlled by means of a passive device that used fiducial markers to track its orientation, and worked similarly to a driving wheel.

Markerless AR games also exist, such as Ewok Rampage [9], that uses Parallel Tracking and Mapping (PTAM) to register the virtual objects over the real world. This game can be played both outdoors and indoors over any flat surface of the real world, and consists in several enemies that appear randomly running towards the player, who has to destroy them.

Other approaches to markerless AR use structured light sensors to detect the environment and light projectors to superimpose virtual elements on the real world, eliminating the need for a display to perceive those virtual elements. Such approaches have been applied to enhance the gaming experience of existing games by projecting additional elements surrounding the game screen [10], or completely immersing the player in a virtual world projected on top of his room [11]. In the later game the player, equipped with a prop gun, has to destroy virtual enemies and avoid traps.

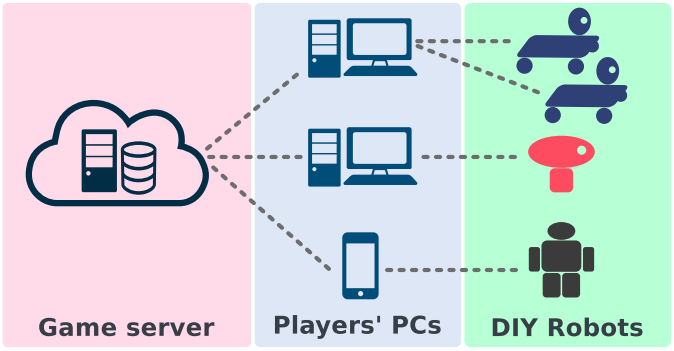

Fig. 2. The system architecture of Robot Devastation with its three main elements

B. Commercial Games

The appearance of more powerful mobile devices such as smartphones and portable game consoles and the fact that most of them incorporate a camera, have raised the amount of commercial AR games available in the market in the last years. As these games are commercial developments, the vast majority of them do not have any publication explaining them or their inner workings apart from their commercial webpage.

Some examples of these games, that usually require one or more fiducial markers to make the augmentation, are Invizimals[1], AR defender[2], Wonderbook[3] or AR Games[4]. The most similar AR game developments to the presented work are the Parrot Drone quadcopter[5] and Sphero [12] games. These games allow a the user to teleoperate the robots from a mobile device, that lays enemies or waypoints (depending on the game) on top of the video stream obtained from the on-board camera. However, these games are usually simple and single or two player games with no multiplayer agenda.

III. System Architecture

The system architecture of Robot Devastation is composed of three main elements: the teleoperated robot, the computer used by the player to control the robot and the game server (Fig. 2).

The current version requires that all these elements are in the same Local Area Network (LAN). In this section these elements will be described in detail.

A. Robot

The main element of Robot Devastation is the robot, which is teleoperated, and acts as an avatar for the player. There should be at least one robot per player, and the minimal robot configuration includes four main elements. A camera is required in order to obtain what the robot is seeing, along

[1] http://invizimals.eu.playstation.com/
[2] http://www.ardefender.com/
[3] http://wonderbook.eu.playstation.com/
[4] http://www.nintendo.com/3ds/ar-cards
[5] http://ardrone2.parrot.com/usa/apps/

with a barebone PC or board for processing and sending the video stream. For communicating with the PC in charge of the robot control, a Wi-Fi link is needed. Finally, the robot should have some way of achieving locomotion.

Two prototypes with different types of movement were developed. The first one is a robot with a pan-tilt mechanism for the camera, that cannot move around the combat zone and simulates a gun turret. The other, a mobile robot with a fixed camera, that simulates a combat vehicle.

These robots run a small program that connects to the player's PC and sends the video stream from the camera while it waits for high-level movement commands (move forwards, move backwards, turn left and turn right). When a movement command arrives, it executes the actions required to perform that command (solving the inverse kinematics of the robot if needed, for instance).

B. Player's PC

The game is played by means of a PC that is connected both to the game server and the robot. This PC displays the augmented robot view on its screen, and waits for user input to control the robot.

An augmented view of the robot's point of view is shown on the display, in which the other robots are marked as targets similarly to a fighter aircraft Head-Up Display (HUD). The robot can shoot virtually, and this shooting is reflected on the user's display according to the type of ammunition/weapon used in the game.

Whenever a target player is hit, its PC sends a message to the server notifying it has inflicted some amount of damage to a player. If the robot suffers enough damage, the game is over for that player, and the server broadcasts this information to the remaining players.

C. Game Server

All the information related to the current game is managed in a central server. This server communicates with the different players, synchronizing the information that each of them keeps about the state of the other players (e.g. health, score, etc).

Each player logs in at the beginning of each game, selecting the team in which his robot will be. Then, the server listens to the messages sent by robots when they hit their robot targets. The health stored is updated by the server for the player that was shot, and the new state of the player that received damage is broadcasted to all the players.

The server is also in charge of logging out a player when it receives a logout request, and informing the players when a robot has lost all its health points and, therefore, has lost the game.

IV. Software Implementation

Details of the current software implementation of the game[6] will be explained in this section. The software is divided into three main parts, one for each of the three main elements in the game's architecture: the robot, the player's PC and the game server.

[6]Currently hosted at https://github.com/asrob-uc3m/robotDevastation

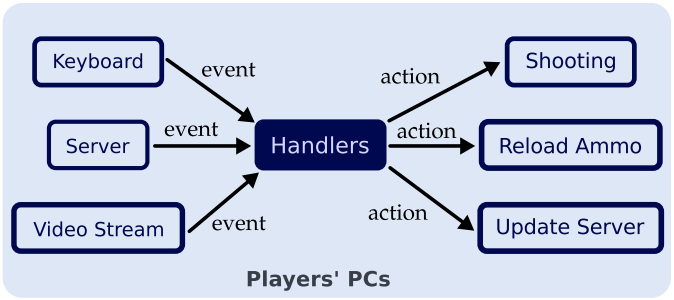

Fig. 3. The game is based on events that are passed to handler functions in charge of executing the different actions.

Since this implementation was a proof of concept of the game, and there was no intention of creating a final product at this point, a game engine such as Unity or Unreal Engine was not used. A much lower level graphics library, SDL, was selected for the (2D) graphics and user input, and YARP [13] was chosen for the communications and video streaming.

The firmware on the robot is very simple, and it is in charge of capturing the frames from the camera and sending them to the player's PC using a YARP port. Another YARP port listens for high-level movement commands such as "move forward" or "turn left", and then these command are interpreted by the firmware, which then executes the actions needed to perform them. For example, for a differential drive mobile robot, a "move forward" command would activate the two motors to propel the robot forward, whereas a "turn left" command would result in the two motors turning in opposite directions to make the robot turn.

The game server is also simple, since it just stores the information about all the players (such as name, id, team, health, score, etc) and updates it according to the game events. Communications are also managed by YARP ports, and a basic API with *login* (adds a player to the server storage), *logout* (removes a player from the server storage) and *targetHit* (modifies the health of a player) commands was implemented.

Most of the computing work is done at player's PC, which runs the most complicated piece of software. The player's PC software is event-based. Some modules, such as the input module or the network module can trigger events and notify other software that an event has occurred. The notified component then acts accordingly to these events. Fig. 3 shows this event-based implementation.

The game flow is controlled with a state machine that first logs in the player in the server in the selected team. When all the players are ready, the game starts, and the program listens to the user's keyboard input (obtained through SDL). If a key is pressed, it will trigger an event that will either send a movement command to the robot, shoot wherever the scope is pointing at, or reloading the weapons. If a target is hit, it will notify the server in order to update the information about the damaged robot.

Since the combat is virtual, elements that are not in the real world must be added to the original image, such as

target marks (with target name and health bar), and firing trails from the bullets or laser weapons. Explosions or other effects when a robot is hit could also be added, although they are not implemented in the current version. Augmentation is completed by adding sound effects for the different actions such as shooting and reloading. The robots have attached QR codes used as fiducial markers, and they are tracked in order to locate and register the different elements of the augmented reality. In the future the game will not need those QR codes, as a markerless AR game is aimed for. Other elements, such as information about the state of the game, and heath and ammunition of the player, are also displayed on the screen (Fig. 4).

All the communications between the player's PC and the game server, or between it and the robot are implemented using YARP ports. An event on the player's PC is triggered by incoming information from the server to process and act according to that information (e.g. the player's robot was hit by another robot and its health has decreased).

V. Examples Of The Robots

The following low-cost robot platforms were built inside the Robotics Society of the University. The platforms were tested within Robot Devastation (Fig. 5).

A. Gun Turret

In order to test the game easily, discarding the effect of the movement of the robot in our tests, a robot that cannot move was designed, as if it were a gun turret. This robot, shown in Fig. 5 in the left part, has a pan-tilt mechanism actuated by two hobby servos. Through this mechanism, even though the robot is static, it can still look around to search other robots and aim at them.

On top of the pan-tilt mechanism there is a Minoru 3D webcam. The servos are controlled with an Arduino Nano board (an ATmega328P-based micro-controller development board), which communicates with a Raspberry Pi (a ARM11-based single-board computer with 512 MB of RAM and running at 700 MHz) in order to receive the movement commands. A Wi-Fi dongle and a LiPo battery are required to make it work wirelessly.

B. ECRO

To exploit the full possibilities of the game, and also to make it more entertaining, a mobile platform, named ECRO, was tested (central robot in Fig. 5).

ECRO is a wheeled platform [14] with two 12 V motors, two motor controllers, an Arduino Mini board to interface the motor controllers with the PC and a netbook PC that sits on top of the robot. The motors are powered from 12 V lead-acid batteries, whereas the netbook is powered by its stock LiPo battery. The webcam of the netbook is used as the robot camera.

C. RD1

The second mobile platform is the RD1 robot, right robot in Fig. 5, a cheap robot that can be 3D printed. It has two hobby servos modified for continuous rotation and a Raspberry Pi board with a Wi-Fi dongle for video streaming and communications with the player's PC. It also has a webcam to obtain the video stream, and is powered with AA batteries (for the motors) and a LiPo battery for the Raspberry Pi.

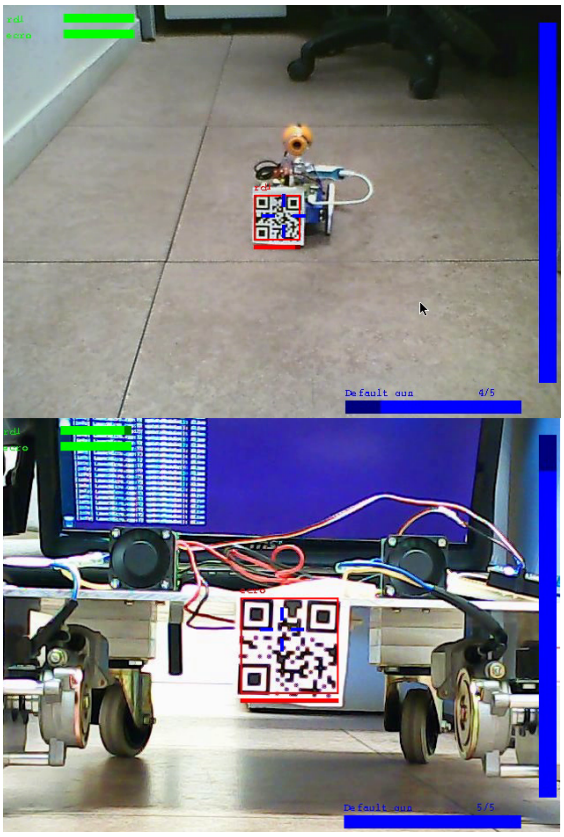

Fig. 4. Screenshots of Robot Devastation gameplay. The resolution of the images the players see is directly given by the robot webcam specifications.

VI. Current Status And Testing

The proposed system was implemented and tested (several video examples[7]). All three robots were assembled and a few games were played. An actual gameplay screenshot can be seen in Fig. 4.

During the testing, some issues were identified. The video stream started to blur with fast maneuvers, an expected (and probably unavoidable) result, as average webcams are not prepared to capture video with fast movements. Moreover, depending on the orientation of the robots, sometimes the QR markers were not detected, so the robot could not hit its target even though the player could see the robot on the screen. This is a source of frustration for the player, specially when he loses the game due to this issues.

Despite this minor challenges to be tackled, some good results can be highlighted. Almost no delay was observed between commands, robot actions and visual feedback, which is important for teleoperating the robot. Also, the modular design of the game allows to add easily new robots (e.g. hexapods, humanoids, etc) as long as they have a camera and

[7] http://youtu.be/MShjf_EqwQg?list=PL0E72403A6C94309E

Fig. 5. Robot platforms tested with Robot Devastation: Gun turret is shown on the left, ECRO is the one in the middle image, and RD1 is the platform on the right.

some way to stream video. It has also possible to extend the game by adding virtual elements such as weapons, HUDs, bonus, etc.

Beyond the current state, some short-term goals can be pointed out. The first one is using a game engine to display more professional-looking graphics. Using a game engine would also probably simplify the software implementation, as many aspects of the game, such as input management will be already implemented and optimized. The other one is to eliminate the need for fiducial markers using, for instance, either visual features, stereo vision or Parallel Tracking and Mapping (PTAM) to track the robots and register them in 3D.

A smartphone version of Robot Devastation is additionally planned as a future development. In this scenario, a robot without camera can be used. As previously seen in Fig. 1, the user directly sees the environment with AR through the smartphone, and the robot's perspective is not used.

VII. Conclusion

Robot Devastation is a multiplayer augmented reality game in the format of a First-Person Shooter. The players' avatar are low-cost Do-It Yourself teleoperated robots. Potential players can build their own robots, and connect to the main server. The architecture of the game allows seamless integration of these platforms.

Augmented reality is a powerful technology for gaming, since it allows the game creators to combine virtual and real world elements, maximizing the interaction between the player and the game, or between the players.

Even though the robots were built with a low budget in mind, they still have a good performance. There is not a significant delay between commands and actions, which could affect to the comfort of the player when teleoperating the robot, and the video stream quality is, in general, acceptable. Issues appear in the video stream when the robot executes a fast maneuver, but that is a common problem when using average webcams.

Acknowledgment

This work was supported by ARCADIA DPI2010-21047-C02-01, funded by CICYT, and RoboCity2030-II-CM (S2009/DPI-1559), funded by Comunidad de Madrid and EU.

References

[1] R. T. Azuma *et al.*, "A survey of augmented reality," *Presence*, vol. 6, no. 4, pp. 355–385, 1997.

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *Computer Graphics and Applications, IEEE*, vol. 21, no. 6, pp. 34–47, 2001.

[3] T. Ohshima, K. Satoh, H. Yamamoto, and H. Tamura, "Ar 2 hockey: A case study of collaborative augmented reality," in *Virtual Reality Annual International Symposium, 1998. Proceedings., IEEE 1998*. IEEE, 1998, pp. 268–275.

[4] W. Piekarski and B. Thomas, "Arquake: the outdoor augmented reality gaming system," *Communications of the ACM*, vol. 45, no. 1, pp. 36–38, 2002.

[5] I. Barakonyi and D. Schmalstieg, "Augmented reality agents in the development pipeline of computer entertainment," in *Entertainment Computing-ICEC 2005*. Springer, 2005, pp. 345–356.

[6] E. Molla and V. Lepetit, "Augmented reality for board games," in *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*. IEEE, 2010, pp. 253–254.

[7] B. Knoerlein, G. Székely, and M. Harders, "Visuo-haptic collaborative augmented reality ping-pong," in *Proceedings of the international conference on Advances in computer entertainment technology*. ACM, 2007, pp. 91–94.

[8] O. Oda, L. J. Lister, S. White, and S. Feiner, "Developing an augmented reality racing game," in *Proceedings of the 2nd international conference on INtelligent TEchnologies for interactive enterTAINment*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 2.

[9] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[10] B. R. Jones, H. Benko, E. Ofek, and A. D. Wilson, "Illumiroom: peripheral projected illusions for interactive experiences," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 869–878.

[11] B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, N. Raghuvanshi, and L. Shapira, "Roomalive: magical experiences enabled by scalable, adaptive projector-camera units," in *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 2014, pp. 637–644.

[12] J. Carroll and F. Polo, "Augmented reality gaming with sphero," in *ACM SIGGRAPH 2013 Mobile*. ACM, 2013, p. 17.

[13] P. Fitzpatrick, G. Metta, and L. Natale, "Towards long-lived robot genes," *Robotics and Autonomous systems*, vol. 56, no. 1, pp. 29–45, 2008.

[14] J. G. Victores, S. Morante, M. González-Fierro, and C. Balaguer, "Augmented reality and social interaction platform through multirobot design," in *Robocity2030 11th Workshop Robots Sociales*, 2013, pp. 131–143.