# A Semi-automatic Approach to Mix Real and Virtual Objects for an Augmented Reality Game

Luiz Maurílio da Silva Maciel, Felipe Andrade Caetano, Rodrigo Luis de Souza da Silva
Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora
Juiz de Fora, Brasil
felipe.caetano@ice.ufjf.br, luiz.maurilio@ice.ufjf.br, rodrigoluis@ice.ufjf.br

***Resumo*—This work describes an approach to mix real and virtual objects in an augmented reality game. The steps to setup the system and the basic logic of the game are detailed. The semi-automatic approach is based on the correct positioning of fiduciary markers over some real objects. An association between these auxiliary markers and one central marker is set during the registration process. Once registered, these markers can be removed from the scene. The main objective of this work is to present this approach in detail, showing how this technique can be used in a augmented reality game based on the interaction between two virtual cannons, and some virtual and real objects.**

***Keywords*—Augmented Reality; Virtual Games; Real Object Registration; Mixing real and virtual objects.**

## I. Introduction

The gaming industry is one of the fastest growing area in recent years. In addition to the search for more realistic graphics, better storylines and gameplay, this growth has prompted the industry to look for new ways of player interaction with the games. The objective is to attract the public from different ages, making interaction more simple and natural, leaving aside the traditional joystick.

Examples of this trend can already be found in the market today. We can cite the Nintendo® Wii Remote Controller ™, the PlayStation3® Move ™, the PlayStation3® Eye ™and Microsoft® Kinect ™.

Augmented Reality (AR) emerges as a new alternative in this scenario, since it allows a very natural interaction of the player without using very sophisticated electronic equipments. Augmented Reality combines virtual computer-generated elements with the real environment and reproduces this combination in a display device such as a monitor, mobile phone, tablet or Head Mounted Display (HMD).

Moreover, basic AR games only need a video capture device, such as a webcam, and a few markers used for projection of virtual objects in the scene. Some games even use real objects as markers.

### A. Related Works

In recent years several games using augmented reality have been proposed. In 2002, a first-person game shooter called ARQuake [10] was proposed. That game is an extension of the desktop game Quake, using augmented reality resources to insert virtual elements in the environment. Markers are used for the placement of objects projected and a tracker system is used to locate the player in the scenario. A similar game called AR Shooter [11] was proposed in 2010. This one uses a gun with embedded video cameras and infrared markers for tracking.

Board games using Augmented Reality has also been developed recently. One of those works uses, as an example, the traditional Monopoly™game [4]. In that game, the board is used as a marker as well as pawns. The board texture is used for tracking so the virtual objects can be projected, increasing user immersion.

Another game that also uses a board to identify the position of the game scenario is the NerdHerder [12]. That mobile game works basically through device movement in the game scenario achieved by the board position.

The ARPong [6] consists of a Pong game using Augmented Reality without fiducial markers. The interaction with the game is made by the hands of the players.

Many works have been done to identify real objects in Augmented Reality systems. One of these works [8] uses a $2D$ marker (a square shaped barcode), that simultaneously identify the real world objects and estimate its coordinate system.

Another paper proposes the recognition of static objects in a room [7]. The geometry of the objects is simple (doors and windows, for example). The system has a priori information about the location and description of the object.

Using a previously calibrated camera it is possible to determine the geometry of relatively complex objects, by selecting some points of the object to be mapped as described in [9].

This paper propose a method for a semi-automatic registration of objects with relatively simple geometry (cylinders and parallelepipeds) through fiducial markers. Once registered, the auxiliary markers can be removed from the scene. This method is described in details in Section II-A.

## II. Setup of the Scenario

The game uses seven fiducial markers. One is positioned approximately in the center of the scene and the position of other objects is determined by its relation to this central marker.

Two markers represent the cannons used by the players to interact with the game. Two additional markers are used for the

representation of virtual objects, one representing a cylinder and one to represent a rectangular parallelepiped object.

The other two markers are used to register the position of the objects in relation to the central marker. The real objects in this prototype are expected to be one cylindrical and the other as a rectangular parallelepiped. They must be placed on the desired positions by the players and the fiducial markers used to the registration of each one must be centered on these objects. Then, the registration of the real objects is ready to be performed, as described in Subsection II-A. Once real objects are registered, the markers that were placed over them can be removed and the game can be started.

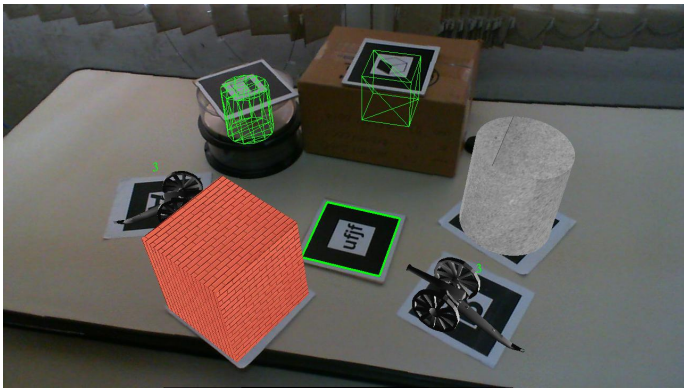
The markers that represent virtual objects should also have their positions set before the start of the game, since during the game, only the markers representing the cannons can be moved.

### A. *Real Objects Registration*

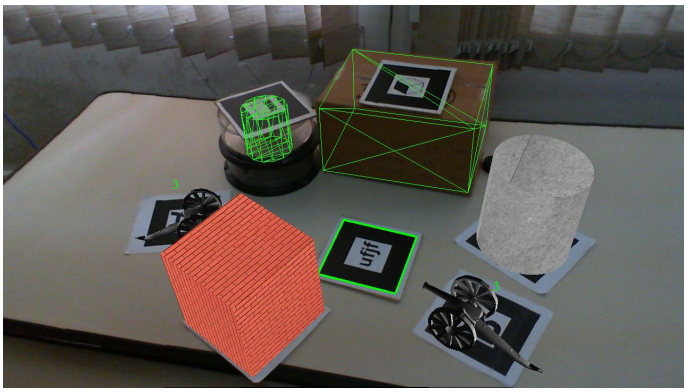
The registration of the real objects begins with the positioning of the markers used for registration above them. The markers must be positioned in the center of the surface of the real objects.

Through the keyboard, the player enters the registration mode. Upon entering that mode, the object's position is computed automatically through the relationship between the registration marker and the central marker. Then using the keyboard, the player can specify the radius, in case of the cylinder, and the length and width, in the case of the parallelepiped. A wired-frame projection with the current size of the object is displayed to assist the user in the registration step.

After confirming the registration, the position and size of the real objects will be saved and the registration markers can be removed from the scene. The Figure 1 shows the steps of the parallelepiped registration while the Figure 2 shows the steps of the cylinder registration.



(a)



(b)

Figura 1: Registration of the parallelepiped with the initial scenario (a) and the final result with the manual width adjustment (b).

## III. AVRLIB

The AVRLib is a object-oriented framework based on the ARToolkit [3] library. The main advantages of this framework over the ARToolkit is the object-oriented interface of the library, in which all of the ARToolkit files were separated in several classes according to their functionality. Most of the code presented in the ARToolkit examples were hidden in the *avrApplication* class.

The class *avrApplication* has methods to define the camera parameters, the threshold for the detection, the single or multi-marker mode and callbacks for events (mouse, keyboard etc). The class also includes a method for obtaining the relationship among the markers of the application.

There is a class called *avrPatt* used to store marker information. The class *avrPatt* contains information like id marker, center, width, transformation matrix and visibility. The class *avrApplication* has a dynamic vector of *avrPatt* and through this method new markers are added to the application, passing a file with the information of the markers to be added, in addition to the associated drawing function.

Thus, to create an Augmented Reality application using AVRLib, it is necessary to include the *avrApplication.h* header and create an instance of the *avrApplication* class.

Just as in the ARToolkit, a drawing function for each marker used in the application must be created. The markers are added in the application by the *addPatt* function. This function receives a configuration file as a parameter, the width and the drawing function for that marker.

The *setCameraFiles* function is responsible for initializing the parameters of the camera through files and the *setKeyCallback* function sets the interaction function through the keyboard (the function must be created). When calling the *start* method, the application starts to run.

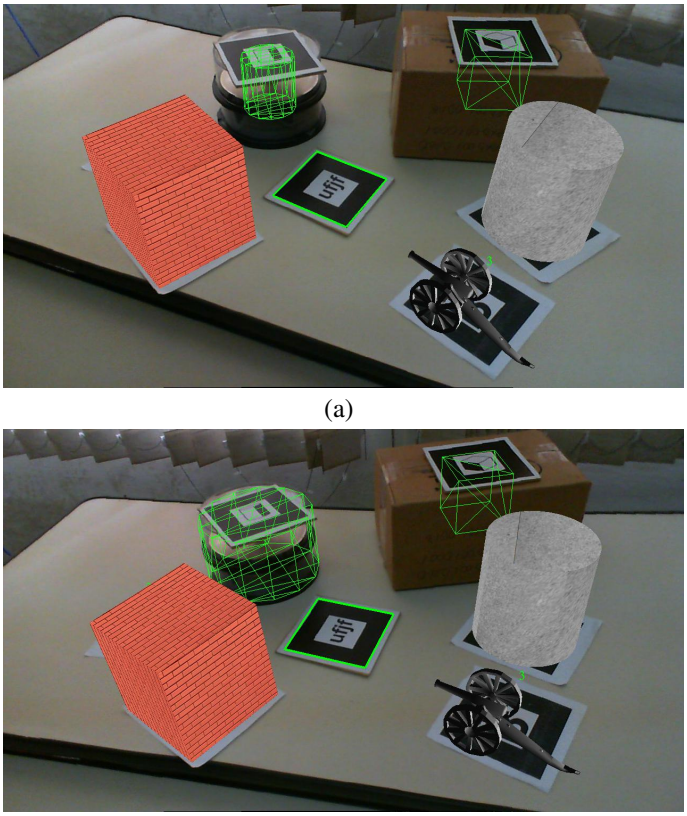
The AVRLib is under development at the Federal University of Juiz de Fora and the first public release is expected to be launched in 2014.

## IV. EXPERIMENTAL RESULTS

This section explains the operation of the game, showing the game rules and collision system developed.

### A. *The Two Hit Game*

The scenario of the game consists of two cannons and four static objects, two real and two virtual. After performing the



(a) (b)

Figura 2: Registration of the cylinder with the initial scenario (a) and the final result with the manual width adjustment (b).

assembly of the scenario as described in Section II the players can start the game.

Each round of the game, one player can move your cannon through the marker that represents it. After setting the position, the player makes the shot using the keyboard. The projectile launched must collide at least into a real object and a virtual object so that it is valid. If the valid projectile hits a cannon, that cannon loses one life. Projectiles that hit a cannon without first hit at least a real object and a virtual object do not decrease the life of the hit cannon.

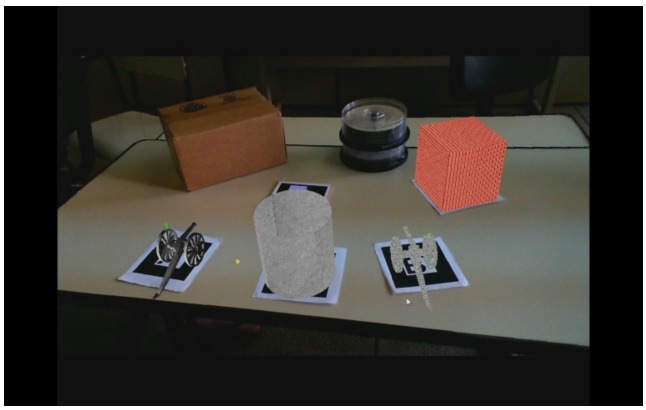


Figura 3: Game in progress

Each cannon initially begins with three lives. As the cannon loses its lives, its texture is changed to indicate the suffering damage. The projectile starts with a black texture when launched and acquires an yellow color to be valid. The Figure 3 shows the game in progress. The projectile has a projected shadow to increase the realism of the game.

### *B. The Collision system*

A simple collision system was developed for this experiment. For simplicity, a system of collision in two dimensions was used by ignoring the height and considering all the objects on a same plane. First, the position of all objects in relation to the central marker that became the origin of the coordinate system is used. Then, the Oriented Bounding Boxes (OBB) of each object in the scene is determined. The collision detection was then made through the bounding boxes obtained using the Separating Axes Method [2].

This method is based on the fact that if the bounding boxes (convex polygons) are not colliding, there is a line that separates them and this line is parallel to one side of at least one of the bounding boxes. Figure 4 shows an example of that line. The line found is parallel to the red side of the yellow arrow bounding box.

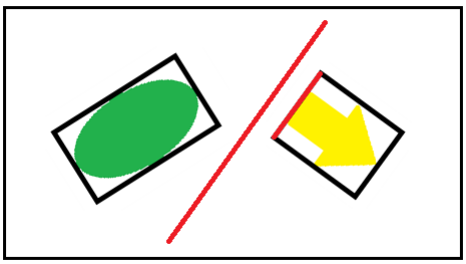


Figura 4: Example of Oriented bounding boxes examples with the Separating Axes Method

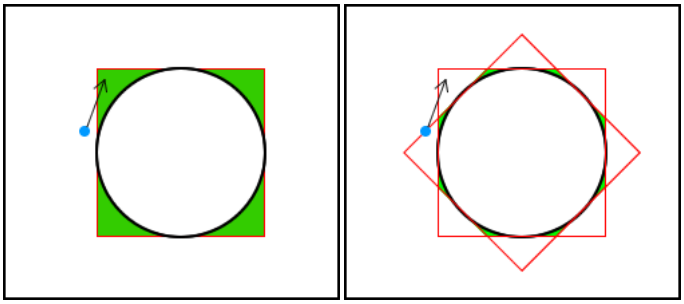
The method consists in determining if this line exists. If it does not exist then the bounding boxes are colliding. The method begins by taking one side of one of the bounding boxes and setting a vector of that line and a vector perpendicular to it.

The next step is the projection of the bounding boxes on this vector. In this step, each vertex is projected onto the vector generating one line with the extreme projected points. If the lines do not overlap then the line that separates the two bounding boxes is found. Otherwise, the other side of the bounding box should be tested. If after testing all sides of the two bounding boxes the line that separates them is not determined, a collision is found.

However, that Bounding Box technique may not be good for detecting collision in some cases. For example, some shots are detected as a collision in the cylinder, when they are not supposed to. The Figure 5.a shows this problem. The green area in the figure represents the area where false-positive detection can occur.

In order to minimize this problem, we use two bounding boxes on the cylinders. The other bounding box is computed rotating the original one with an angle of 45°. The Figure 5.b

shows these bounding boxes. In this case, there is a collision only when a collision with both bounding boxes is detected. We can observe in this figure that the false-positives area is decreased.



(a) (b)

Figura 5: False-positive detection in the cylinder bounding box (a) and two bounding boxes to minimize the false-positive detection problem (b).

Only one projectile (a sphere) moves during a shot. In order to optimize the calculations, initially we calculate the distance between the sphere and all objects and the collision test is done only with the nearest object.

Once a collision is detected, the new direction of the sphere motion is determined. That direction must be such that the angle of reflection is equal to the angle of incidence (Figure 6).

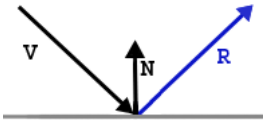



Figura 6: Reflection of the direction [1]

We used the following formula to calculate the new direction [1]:

$$\vec{V}_{new} = -2 * (\vec{V} \cdot \vec{N}) + \vec{V} \quad (1)$$

where $\vec{V}$ is the direction before the collision, $\vec{N}$ is the surface normal where the collision occurred and $\vec{V}_{new}$ is the direction after the collision. Based on the Figure 6, Figure 7 shows how the Equation 1 is obtained.

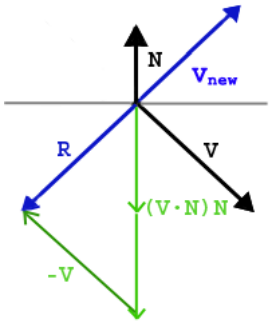



Figura 7: Computing the new direction of the projectile.

The simplicity of the presented collision system allows the entire system to run at 30 frames per second on a Intel® Core™2 Quad Q9550, 4GB of RAM, NVIDIA® GeForce™GTS 250 equipment.

## V. Conclusions and Future Works

This paper presented a game using Augmented Reality and iteration with real and virtual objects. The position and dimensions of the real objects are obtained semi-automatically by the relationship between markers. This prototype was developed at the end of an augmented reality postgraduate course and the main focus was on the interaction of real and virtual objects.

Some problems were encountered for detecting the height of real objects. Some of these problems results from distortions in the markers detection by the AR libray. Furthermore, the relationship of height between the markers is not always accurate, since any rotation in $X$ or $Y$ axes between the central marker and the marker used in the calibration, may interfere in the ability to determine where is the base plane of the game.

For future works, techniques for automatic detection of real objects can be used, making easier and more accurate the configuration and identification of objects. Also, the accuracy of the collision system can be improved by implementing a system based on Octrees [5].

## Referências

[1] 3DKingdoms. http://www.3dkingdoms.com/weekly/weekly.php?a=2, Jan. 2006.

[2] D. Eberly. Intersection of convex objects: The method of separating axes. http://www.geometrictools.com/Documentation/MethodOfSepa ratingAxes.pdf.

[3] H. Kato. *ARToolKit 2.33 Documentation (Alpha Version)*. Human Interface Technology Laboratory, University of Washington, http://www.hitl.washington.edu/artoolkit/documentation/, 2005.

[4] E. Molla and V. Lepetit. Augmented reality for board games. In *9th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2010, Seoul, Korea, 13-16 October 2010*, pages 253–254. IEEE, 2010.

[5] M. Moore and J. Wilhelms. Collision detection and response for computer animation. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 289–298, New York, NY, USA, 1988. ACM.

[6] Y. L. B. Nogueira, M. O. Freitas, C. A. Vidal, and J. B. Cavalcante-Neto. Arpong: A viability case study of markerless ar-based interface to computer games with webcam. In *IX SBGames - Florianópolis - SC, November 8th-10th, 2010*. SBC, 2010.

[7] C. Pereira. Object recognition in augmented reality. http://www.umcs.maine.edu/ markov/chesterslides.pdf, Dec. 2002.

[8] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *APCHI*, pages 63–69. IEEE Computer Society, 1998.

[9] E. Rose, D. Breen, K. H. Ahlers, C. Crampton, M. Tuceryan, R. Whitaker, and D. Greer. Annotating real-world objects using augmented reality. Technical report, EUROPEAN COMPUTER-INDUSTRY RESEARCH CENTER GMBH, ARABELLASTRASSE 17 D-81925, 1995.

[10] B. Thomas, B. Close, J. Donoghue, J. Squires, P. D. Bondi, and W. Piekarski. First person indoor/outdoor augmented reality application: Arquake. *Personal Ubiquitous Comput.*, 6(1):75–86, Jan. 2002.

[11] D. Weng, D. Li, W. Xu, Y. Liu, and Y. Wang. Ar shooter: An augmented reality shooting game system. In *ISMAR*, page 311. IEEE, 2010.

[12] Y. Xu, S. Mendenhall, V. Ha, P. Tillery, and J. Cohen. Herding nerds on your table: Nerdherder, a mobile augmented reality game. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*, CHI EA '12, pages 1351–1356, New York, NY, USA, 2012. ACM.