Real Time Lighting

André Pereira pg19828

Rafael Silva pg20202

UCE of Computer Graphics

Department of Informatics

University of Minho

# Abstract

Recycling is, due to many factors, an important part of any sustainable society and because of such importance, it must be an important piece of the education. In this report will be presented an augmented reality application, which objective is to teach young children the importance of recycling, through an entertaining game. The technical issues of the application will be presented. However, the impact of its purpose was not validated.

# Introduction

A common problem is that most of human activities produce waste, which we commonly call trash. The way trash is handled is one emerging issue to the global community. It is usually taken to sanitary landfills where they are accumulated, polluting the environment and decreasing our life quality for the future. Aiming to reduce the environmental impact, the 3R's policy was created, which proposes that the population should Reduce the amount of waste it produces (e.g. use of products with bigger packages to minimize the amount of packaging waste), Reuse resources (e. g. reusing supermarket bags for packaging trash), and for last and most important, the context of this project: Recycle, i.e., reuse a material as feedstock to another. For example, in the city of Oporto and its metropolitan area, 488 216.35 tons of waste were produced in 2011, of those 53796.73 tons were recycled [1], helping to save the environment.

For these recycling policies be viable it is necessary that the population is properly educated, which currently happens from the children early years. To help this education we propose a game that promotes recycling as something positive. The game is an augmented reality application that aims to teach young children the importance of recycling, as well as helps their familiarization with the future of interactive technologies.

# Concept And Purpose

Our objective was to better educate children regarding the environment protection. Was opted to develop an application which purpose is to teach, in a recreational way, how to recycle and instigate these practices in their life. Other benefit of this approach is the familiarization of the children with the new technologies, namely augmented reality, starting in their early years.

The application developed implement augmented reality methods and computer vision algorithms in order to combine the real world, captured by a standard camera, with virtual geometry, presenting the final combination on the screen. Its purpose is to be a game, in which is displayed a set of objects moving on the screen space. Each object belongs to one of the three recycling categories or undifferentiated trash. The game is for four players and each has a different marker with an associated recycling category, and the camera records their position. One point is awarded if a player intersects its marker with one of the respective objects on the screen. If it intersects with an object of any other category one point is taken. When all objects are intersected, the player with the highest score wins.

# Technology Used

The main difficulty on developing the application is to overlay the virtual geometry with the real world, captured by a camera. ARToolkit [2] is an Open Source library that makes the development of augmented reality applications easier.

ARToolkit allows the overlaying the real and virtual worlds, later displayed on the screen, using predefined patterns to track the users viewpoint. These patterns, captured by the camera and recognized by the library, allow the placement of virtual geometry relatively to them. The library uses computer vision algorithms to calculate the camera position and orientation to the patterns.

It is easy to integrate ARToolkit with OpenGL [3]. It provides the *ModelView* matrix to place the geometry on the patterns. After getting the matrix it is possible to do any transformation in order to change the relative position of the geometry – it only acts as a reference point. It is possible to detect several patterns as the same time and handle each *ModelView* matrix independently.

The rendering pipeline has 5 stages:

* **Frame capture**: ARToolkit primitives to capture a frame from the camera;
* **Frame display**: Display the captured frame on the screen;
* **Pattern detection**: Compare the detected patterns from ARToolkit with the predefined in the application. It is possible to detect many patterns in a single frame;
* **Get the transformation matrices**: Get the respective *ModelView* matrix from each pattern detected;
* **Draw virtual geometry**: Draw the virtual geometry, possibly using the matrices from the patterns, and display it on the screen, overlaying the previous displayed frame.

All the stages can be modified but only the fifth will suffer major alterations, depending on the application and desired output. This pipeline is repeated for the render of every frame. The frames per second that can be rendered are dependent on the frames per second that the camera can capture.

Figure 4.1 – Rendering pipeline of the application.

# Implementation

As presented in the previous section, the render of each frame is divided in five steps. They will all be presented, although the first four are almost standard for every application.

This application requires the implementation of some key features:

* **Multi-pattern recognition**: three patterns, one per each recycling category;
* **Object drawing**: drawing the different objects in the screen space, regardless of the pattern positions;
* **Intersection of the patterns with the objects**: since they are in different spaces (*ModelView* vs *Projection*) accurate intersection detection is hard to achieve.

Each of these tasks will be explained in the respective stage of the pipeline.

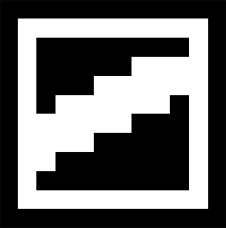


Figure 5.1 – Example of two patterns detected by ARToolkit.

**Implementation**

First it is needed to load the object models and patterns to detect. This is done in the init function, which is called in the main. To load the models is used the *glm* library, developed by Nate Robins. It will not be further explained, as it is not in the scope of this report. The patterns are loaded using the arLoadPatt primitive.

Then the video capture must be initialized. Note that the camera must be calibrated to obtain the best results, however for this application such accuracy is not required. At the time that the camera is initialized the display window is also created. Then the video capture can start. This is done by the arVideoCapStart primitive. Finally, the main loop must be initialized, much like with OpenGL, with argMainLoop, where is passed the loop function, which is called in every frame. Is inside this function that the rendering pipeline takes place.

The first stage of the pipeline is the capture of a frame by the camera. This is done by the arVideoGetImage primitive. This function returns an ARUint8 data type, which is an array of unsigned chars with 8 bits, holding the image data. If the function is unable to capture the video input, the application exits.

The captured frame is then displayed on the application window. At this phase, the displayed image is just the frame captured by the camera, i.e., with no virtual geometry added. The argDispImage displays the image. Note that before the displaying the draw mode must be set to 2D, by the argDrawMode2D function, as the image is a 2D entity.

The third stage is the marker detection. The arDetectMarker function returns an array of ARMarkerInfo, which is a list with all the patterns detected on the frame. Note that it is all of the predefined patterns in the library. The elements of this list must then be compared to the patterns loaded during the application initialization. If a match is found the handleDet function is called.

The handleDet function handles the detection of a pattern. It receives the ARMarkerInfo structure, which holds the information about the pattern, and, using the arGetTransMat, it extracts the *ModelView* matrix of the pattern – the fourth stage of the pipeline. The matrix is added to a global data structure so it can be later used to draw the virtual geometry.

The final pipeline stage is divided in two steps. The first is the drawing of the moving objects on the screen. This is done by the draw\_objs function. The drawing must be done in the projection space, so it is not affected by the movement of the camera relatively to the patterns (and vice-versa). The captured image now acts as a background to the whole scene. The random movement of the objects is calculated just before they are drawn.

The second step is the drawing of the circles on the pattern positions. The objective is just to give a better feel of the position of the patterns in the virtual space, making the intersection of them with the moving objects easier to achieve by the players. Also, during this step the intersections are determined. For each pattern is called the handle\_objs function, which, based on the *ModelView* matrix of the pattern and the objects, traces a vector between the pattern and each object and determines its magnitude. This value is the distance between the pattern and the object. If it is lower than a given threshold it is considered that an intersection occurs and the score is updated. Also the intersected object will not be drawn anymore.

To add a nicer visual effect it is recorded a one second history of the positions of each pattern, on a global data structure. During the previously explained fifth pipeline stage, when the circle is drawn on the current pattern position it is also drawn on every position in the data structure holding the history of the pattern positions. During this phase if there is a position older than one second it is removed from the data structure.

**Implementation Difficulties**

There were two main difficulties in this implementation. The first was to draw the objects in the projection space. It had to be ensured that, during they movement, they would not go out of the displayed area. The strategy used was to consider boundaries, determined by experimentation, that the objects cannot pass. The size of the object can affect if it is totally on the screen or partially out, depending on the boundaries used.

Since the pattern and object positions are in different spaces (*ModelView* vs *Projection*), the test of the intersection is more complex. There was also considered boundaries for the *ModelView* space, delimiting the area that the screen shows, at a certain distance of the patterns to the camera. These boundaries were determined by experimentation. Then, the pattern coordinates are transformed from the *ModelView* space to the Projection space, based on the boundaries. This approach provides plausible results, however if the distance of the pattern to the camera is varied it lacks precision.

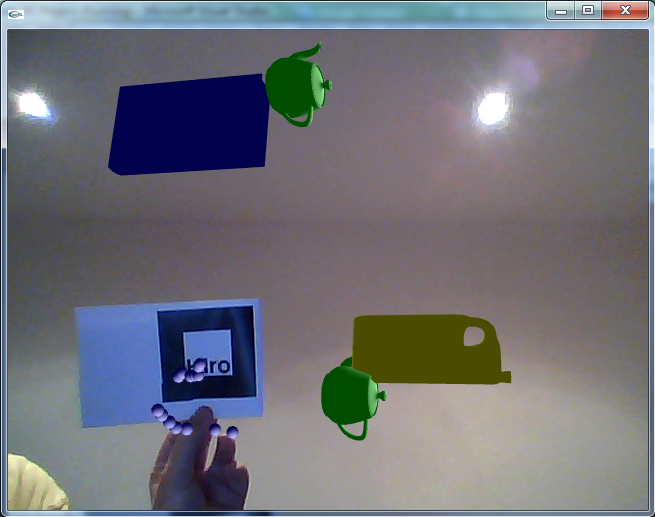


Figure 5.2 – Screen shot of the developed application. Note the objects on screen space and the trail of spheres on the pattern positions.

# Conclusion

In this report we presented an educational game, which objective is to teach children the habit of recycling. It is an augmented reality application where the players must intersect objects on the screen with patterns that they hold in order to win the game. Allied to the education of recycling, it provides an entertaining way to familiarize children with the current technologies.

The ARToolkit was used to provide some augmented reality features required for the application. This library offers an easy interface to operate with the camera, which allows the window creation, capture of frames by the camera and identification of patterns in the image. Along with the pattern identification, the library provides its *ModelView* matrix. These patterns act as reference points in order to place virtual geometry overlaying the image captured by the camera, obtaining an augmented reality scene.

While developing the application we encountered some difficulties. One of the most important is related to the detection of the intersection of the objects moving on screen space (*Projection* space) with the patterns (*ModelView* space). The distance of pattern to the camera affects the detection, as it may vary its *xx* and *yy* coordinates. When transforming these coordinates to the *Projection* space this deviation will be increased and the compared coordinates do not reflect the actual 2D position of the pattern.

As future work, the mentioned problem can be corrected and the validity of the application as an educational tool tested in elementary schools.

References

1. LIPOR – Serviço Intermunicipalizado de Gestão de Resíduos do Grande Porto, *Relatório de Análise Estatística LIPOR*, Visited in: 10-06-2012.
2. Philip Lamb, “ARToolKit”, Available at: http://www.hitl.washington.edu/artoolkit/, Visited in 2012/5/15.
3. Khronos Group, “OpenGL”, Available at: http://www.opengl.org/, Visited in 2012/5/21.