Interactive Augmented Reality Card Game

Vince Bui
University of Ontario Institute of Technology 2000 Simcoe St. N Oshawa, On
vincent.bui@mycampus.uoit.ca

Sing Chen
University of Ontario Institute of Technology 2000 Simcoe St. N Oshawa, On
jia.chen@mycampus.uoit.ca

Andrew Milner
University of Ontario Institute of Technology 2000 Simcoe St. N Oshawa, On
andrew.milner@mycampus.uoit.ca

Richard Pitul
University of Ontario Institute of Technology 2000 Simcoe St. N Oshawa, On
richard.pitul@mycampus.uoit.ca

ABSTRACT

In this paper, we discuss the creation of an augmented reality card game built using the ARToolKit. Augmented reality offers a unique environment in which one has the ability to learn. Skills imparted by augmented reality environment include interpretation, multimodal thinking, problem solving, information management, teamwork and flexibility [1]. In an attempt to stimulate these skills we have created an interactive card game in which the participant must manage, observe and strategically place playing cards in order to defeat their opponent. We hope that the prototype simulated environment we have created will eventually evolve into a learning tool which can be used by students to impart new skills and train existing ones.

Categories and Subject Descriptors

I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism --- Virtual reality

General Terms

Algorithms, Human Factors.

Keywords

Augmented reality, video object tracking, surface marker, interactive learning environment, critical thinking.

1. INTRODUCTION

Finding new and innovative ways to engage students in the classroom is essential to the future of education. Augmented reality (AR) provides a unique way to capture and engage a student in the classroom environment. AR allows students to interact in engaging and authentic environments for learning, activating, and practicing skills such as problem solving, information gathering and management, and teamwork [1]. The purpose of our simulation is to test out the capabilities of the AR system. We accomplished this test of our system by creating an environment in which two monsters battle against each other and are commanded by spoken word commands. Each player in the simulation takes a turn in which they can attack the opposing player. When the opposing monster's health reaches 0, the game is over. Figure 1 depicts the virtual environment with characters and arena.

Figure 1 – Virtual Environment

In this paper we discuss the initial creation of the simulated environment which will eventually be used as an interactive learning tool for students. First we will discuss some fundamentals of augmented reality games and how they can be used as learning tools for today's students. Next we will discuss the methods used to create the virtual environment. These include the use of the ARToolKit library for the handling of virtual object placement in the real world environment, the use of 3D sound to immerse the user in the virtual experience, the use of voice recognition as an input device, the creation of the avatars, and the FBX SDK which is used to render our virtual

world. This section is followed by our conclusions on the initial phase of our project, as well as plans for future iterations of our project.

2. BACKGROUND

Augmented reality games have a distinct difference to regular games. In the past we had games which would be played in the real world. Eventually computer games were created but for a long time there was a divide between the two types. Unlike these games, augmented reality games or AR games for short attempt to bridge the gap and involve the player in all aspects. The main intention of AR games is to use new media forms in such a way that they integrate with the real world. This can take many forms, from using Geo Position Devices to adding virtual objects into a real world environment.

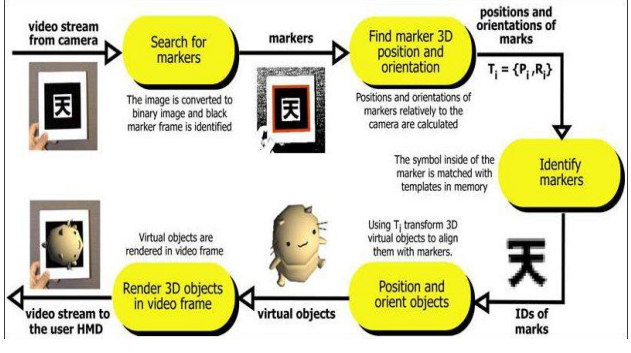
In the past some attempts at using AR type environments in learning situations have already been done. For instance BMW has developed glasses which superimpose images onto different parts of the car's engine and even direct a mechanic as to the steps to take in order to complete certain tasks. AR allows the creators to provide a unique perspective to the users, as it allows for the unique possibility that a user has the ability to walk around the outside of the game. In this way the user must make an active choice to suspend their disbelief and take part in the game rather than being forced into it as other games currently do. This choice is what makes games that use this technology much more beneficial, if the user knows ahead of time that the games intention is to teach them, and they still make that decision to play the game they become that much more apt to learning as they have actively chosen to learn by accepting their role.

3. METHODS/PROCEDURES

3.1 The ARToolKit

The ARToolKit is an augmented reality library which we used to generate the virtual interface of our simulated environment. The library functions by using computer vision algorithms to generate 3D objects onto real world markers. To do this the computer system needs to know where the viewer is looking in the real world in order to draw the object in the proper perspective. The camera viewport position and orientation is determined with the aid of a unique physical tracker and computer vision algorithms. This calculation is accomplished in real time by the software and is the most difficult part in the augmented reality process, because the virtual object must align with the real world.

Figure 2 – Video Capture Process



The tracking system used by the ARToolKit, depicted in figure 2, functions as follows. The camera first captures a real world image and then sends the image to the computer system to be processed. The software searches the captured image for any identifiable square shapes. If at this point a square marker is found the software will calculate the distance and orientation of the camera relative to the position of the square. The model is then drawn on top of the captured image of the real world so that it appears to be on top of the square marker. Finally, the image is output to the user display so when the image is viewed, the virtual object appears to be in the real world. The marker detection algorithm is based on corner detection with a fast pose estimation algorithm.

Figure 3 – Threshold Image

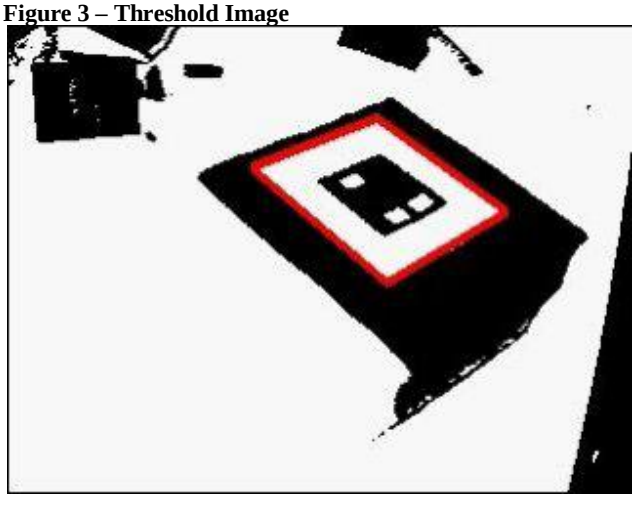


Figure 4 – Interactive Object Overlay

Figure 3 shows the threshold image used by the software to detect the marker. The red square around the marker shows that the marker has been found. Once the marker has been found the virtual object can then be oriented to match the marker and drawn on top as shown in figure 4.

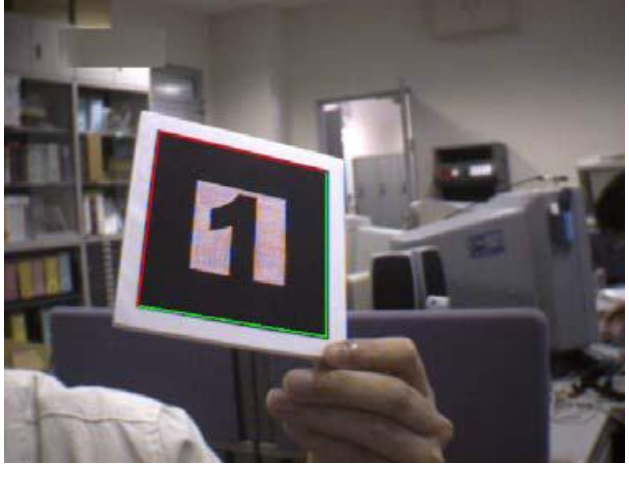
The software can also be trained to identify user created markers. This ability was utilized in our simulated environment to add to the experience of the user, and also to serve as a means of identifying which monster character would

be summoned when a card is played. Figure 5 depicts a sample of one of our user defined markers.

Figure 5 – User Defined Marker

The training program works by running the mk_patt program which comes with the ARToolKit, and then orienting the marker so the top left corner of the marker is red and the bottom left corner is green as shown in figure 6. At this point the image was captured and the software performed a training operation so that the pattern could be recognized when displayed to the camera.

Figure 6 – Training System Process



However, there are several limitations which can make the identifying of markers difficult or impossible by the software.

The unique marker used by the ARToolKit must be visible and unobstructed from the viewer's field of view. If any part of the marker is blocked the computer vision algorithms will be unable to calculate the perspective of the virtual avatar. The size of the marker also affects the range at which the camera can recognize the object. Table 1 shows a list of different sizes of markers and their estimated effectiveness range.

Table 1 – Pattern Size Vs. Usage Range

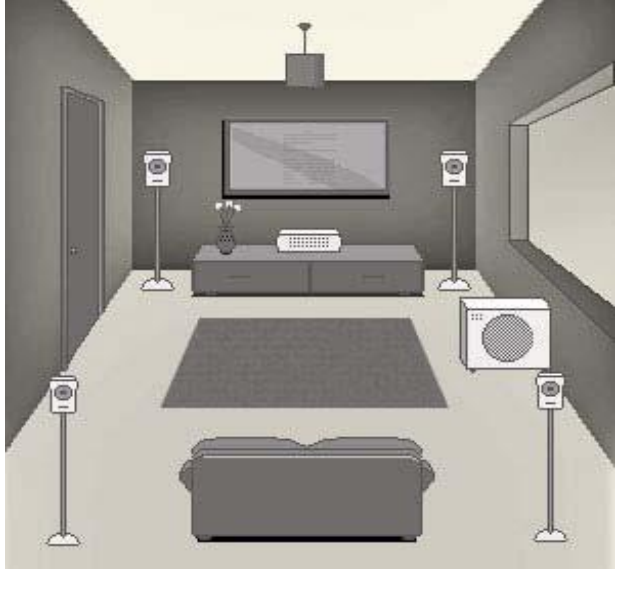
Pattern Size (inches)	**Usable Range** (inches)
2.75	16
3.50	25
4.25	34
7.37	50

Additionally, the complexity of the inner pattern can shorten the range of the camera as the marker's image becomes more complex. If the marker at any point becomes bent the software system has difficulty calculating the orientation of the virtual object. This is because the marker no longer has a perfect square surrounding it. This negatively effects the corner detection algorithm and can cause the virtual object to suffer from jitter. As position of the camera moves closer to the horizontal plane the marker is on, recognition of patterns becomes more difficult for the computer vision algorithms. To avoid this problem the camera should remain at a distance above the marker. Lighting can also be an issue when trying to identify a unique marker. If lighting is of poor quality the software will be unable to recognize the marker and will not draw the virtual object. However, threshold values can be altered in an attempt to compensate for low light levels. This increase can have the negative effect of recognizing non-marker objects as markers, and will attempt to draw the virtual object at that location. Additionally, if the light source is too bright the markers may experience a glaring effect. This will also prevent the software from being able to recognize the pattern.

3.2 3D Sound Environment

In general 3D sound in a simulated environment takes the position of the listener and translates how a sound is heard in a relative manner. In this way a sound from an entity to the left of you would come from the left speakers. Meaning in the usual case the listener or listeners are assumed to be within the sound environment, as scene in figure 7.

Figure 7 – Typical Speaker Setup



The idea being that the person is in the simulated environment and the sounds are being played around them. In our simulated environment we have a unique difference in that the spectators are watching the simulated environment outside of the enclosed space. The sounds in this situation are instead happening in a localized environment and must be played in a fashion unlike the general case. Using the FMOD sound system (a commercial audio library made by Firelight Technologies) we found that the effect of having the sound translated based on the listeners relative position created side effects which served to detract from the experience. Refer again to scenario in figure 7, if one is sitting on the couch as a listener and an object was intended to be forward and to the right of that position it would be played on the front right speaker at an intensity level based upon how far away the sound object is in relation to the listener's relative position. However this approach of representing three dimensional sounds does not work in our intended simulated environment. The problem we encountered occurs when we have another listener at the other side of the room. The sound should be played louder, in their perspective on the back left speaker. This type of sound environment does not work in our intended simulated environment. The volume of the sound at each speaker should not be affected by its distance from the user; rather it should be solely affected based on its location in the environment and its distance from each individual speaker. All changes in volume are based on the distance of the object to the individual speaker; the user should then be able to perceive where in the environment the object is emitting sound from regardless of the user's position in the room.

Figure 8 – Sound Emanating Positions

Figure 8 represents the potential layout in which the four speakers surround the simulated environment's playing area. Assume that the red and blue chips each are the source of an emitted sound. In this situation the red chip's sound should appear to be loudest on the far left speaker, the one which it is closest to based on its distance to the speaker. Likewise the blue chip should play loudest on the speaker it is closest to; this should remain constant regardless of the user's location in the room. In this situation the sounds should appear to emanate from the position of the blue and red chips on the table. The sounds are being simulated as coming from their position on the table, and if the user were to move further away from the desk, the volume would naturally get quieter in relation to their distance away from the table.

However, in the FMOD sound system the native 3D sound options are incapable of simulating this particular environmental effect. To compensate for this shortcoming a change in volume on a per speaker basis for each individual sound is required. Basically if an object is within the listening distance of a particular speaker the sound is played on that speaker which it is closest to.

Figure 9 – Listening Radius of Speaker

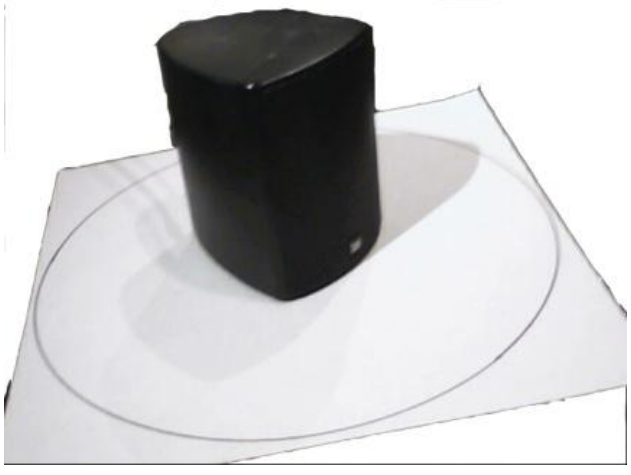


Figure 9 shows a speaker with a listening radius represented by a white disc. In our simulated environment each speaker's radius is large enough so that it overlaps partially with the other speakers' listening radiuses. This ensures that no point in the simulated environment will be without some listening space. If more speakers are added around the volume of the listening area smaller listening radiuses and a more precise estimation of the sounds position can be achieved.

3.3 Voice Recognition Input

In order to increase the immersive experience for the user we chose to forego the use of traditional input devices such as a gamepad and instead implement a voice recognition system to interact with the simulated environment. The voice commands used are similar to those used on the Pokémon television show to increase the realism of the experience. Voice commands are captured and then commit different operations depending on the words spoken by the user. This was determined to be an effective way to immerse a participant in the simulated environment, allowing them to take a more active role as the simulation progresses. Each user has the ability to literally tell their avatar what to do.

Using the Microsoft Speech SDK we were able to accomplish voice recognition simply, however one important consequence exists in our simulated environment. Accepting voice commands from each player at the same time is not easily dealt with. Since the commands being used are short we are able achieve speech recognition without the long process of training the voice recognition software. The inherent flaw with this type of input device is that one person has the ability to control the other user's actions as well as their own when playing the

simulation in the same venue together. This means that an understanding must occur between players so that they take turns issuing their commands instead of attempting to simultaneously give commands to their avatars. In order to move from a turn based system to a real time simulation some training beforehand can be completed so that the voice recognition system can differentiate between the different voices.

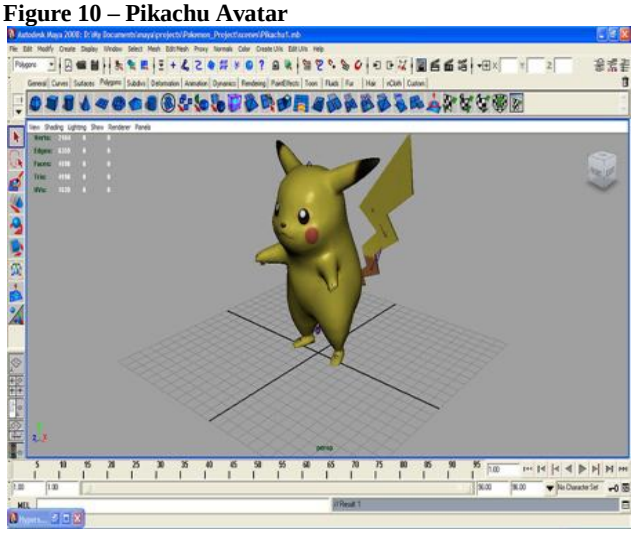
The Microsoft Speech SDK (SAPI for short) hooks into the speech to text services which comes preinstalled on each Windows operating system. This means that any training, if we were apt to do it, would be handled by the operating system. However the drawback to using this system is only one user profile can be in use at any given time. This means that we would need to keep commands short and fairly distinct in order to provide a means of distinguishing between user actions in the multiplayer simulated environment. In our program we initialize the SAPI system and supply it with a file (written in the XML format) that contains the list of words or phrases of interest. When our program is running, the Windows message queue must be continuously checked for a certain message. A message the SAPI system outputs when a recognition event has occurred. Contained in this event is an ID for the command that has been recognized. Based on this ID our program completes the task associated with that specific command. For example, if the word "Defend" was supplied in the XML file, when one spoke the word "Defend" the SAPI system first recognizes the spoken command and then sends out a "Recognition Event Occurred" message, at which point our program determines the particular ID of the event and commits whatever operation the "Defend" command is associated with.

3.4 Creating the Avatars

The process of modeling the creatures for the virtual environment required several steps in terms of getting the models, creating different animations for each creature and then finally exporting the object into a specified file format so that it can be loaded into the system.

The models were acquired from Super Smash Brothers Brawl, a game produced and developed by Nintendo. The reason these specific models were used was because of availability and these creatures were part of the simulated universe we are attempting to create. The two models are based on two fictitious creatures known as Pokémon. Each creature was given a set of animations: an attack, defend and death animation.

Figure 10 – Pikachu Avatar



To create the attack animations for Pikachu (figure 10), the animator conducted research on the subject of how rodents move in their environment and defend against other animals. Also noting that the game involved Pokémon creatures, the animator exaggerated certain aspects into the attack animations to make it more theatrical to the audience. For the defense animation, the animator chose to incorporate a basic rodent defense consisting of keeping the head down and raising the hind legs to make the creature appear larger to predators. The death animation was developed from actual footage of the Pokémon animated television series.

Figure 11 – Charizard Avatar

For animations of the Charizard creature (figure 11), research was completed on the movements of lizards. Additional information about the character was gathered from watch the Pokémon television show. For attack animation the animator observed the flamethrower attack used by the character on the show as a base for creating the creature's movements. The

defend animation was inspired by birds in which they flap their wings to defend themselves. Additionally Charizard shields his body with his wings to add a more dramatic effect. For the death animation, the animation that fit best was similar to that of the Pikachu model. Charizard would simply flop on the floor with his wings covering his body.

Once all the animations were created and cleaned up, the model were exported in the FBX format for use in the simulated environment.

3.5 The FBX Format

The model and animations engine used in our simulated environment has the ability to support two file types [sic] FiLMBOX (FBX) or OBJ models. However, the OBJ file format is incapable of handling 3D model animations, where the FBX file format has this ability. It was for this reason we chose to use the FBX format. The FBX file format is owned by Autodesk and originally created by Kaydara. The FBX software development kit is made freely available by Autodesk. The code written to load and handle FBX objects was largely based upon examples provided with the Autodesk SDK, but was developed to work with unique requirements of our simulation.

In order to load models into our simulated environment the following code elements are required. First the FBX SDK requires a manager to be initialized. The KFbxSdkManager type allows for models to be held and managed in memory. Secondly, the actual FBX file must be loaded into the simulated environment. This is accomplished using the KFbxImporter command. This command parses the FBX and checks for any potential errors in the file's structure. If an error exists or the format is inconsistent, the model is disregarded. The FBX file may be left as a binary file or an ascii file, both types can be parsed by the parser. In this project specifically we dealt with ascii FBX files. As ascii file types it was easier for us to manipulate the files manually, in our case it allowed us to put in multiple takes into one file. This allowed us to save memory overall in the program since we did not need to initialize a new scene for each animation. Finally, the KFbxScene command is called for each scene one wishes to generate in the simulated environment. Scenes contain information about the how the model is drawn, and also stores any model animation if it is available. Each scene also requires information about the specific model's number of takes (a take is an entire animation loop; an example would be a run take, walk take, etc.), the number of poses made by the model, and the number of textures used by the specific model. If a scene contains animation information the FBX file will contain additional information not present in static FBX files. This information includes, the period (how long the animation is), the start frame, the end frame, and the current frame being executed. Functions are available to alter the play speed of each animation if other animations do not appear to play together at consistent speeds.

The process to load each model using the FBX SDK is as follows. First the FBX parser checks the file format for any errors and consistency. A scene is then created based upon that file, and the axis system is altered to accommodate for a right handed Cartesian system. Each model is then normalized, and the NURBS in the scene are converted to a polygon mesh in order to be drawn correctly. The point cache data, takes, poses, and textures are all stored in arrays. If no animation data is included with the FBX file no further steps are required, however if animations are included the playback time, the current take, and the start and end frames are acquired. To switch between different animation takes, the current scene's take buffer must be changed to the desired animation, as well as the start and end frames of the new take.

While developing with this format we came to realize two major issues with the technology. First, the FBX format is not capable of handling multiple texture image files. This proved to be an issue in our simulated environment as many of our avatars were built with multiple texture image files. The second issue we encountered involves additional model geometry not connected to the main body of the object. All the additional model geometry was subsequently disregarded.

4. CONCLUSIONS

This paper described the initial process of creating the virtual environment and the ongoing process which will eventually be used to impart learning to students. In future iterations of the project we hope to further develop the virtual world to include additional avatars, worlds and abilities. Once the virtual environment has been completed we hope to test our virtual simulation as a learning tool in a classroom environment to determine if it is an effective learning tool which meets the needs of today's students.

5. ACKNOWLEDGMENTS

We would like to thank Bill Kapralos for motivating us to create a virtual reality project using augmented reality which focused on serious games, and we would like to extend a thanks to Nintendo for the use of the avatar models.

6. REFERENCES

[1] Schrier, K. Using augmented reality games to teach 21st century skills. ACM. New York, NY. USA, 2006.

[2] Lyu, M.R., King, I., Wong, T.T., Yau, E. & Chan, P.W. ARCADE: Augmented Reality Computing Arena for Digital Entertainment. Aerospace Conference, 2005 IEEE., 2005.

[3] FMOD. Music and Sound effects system. Firelight Technologies Pty, Ltd. www.fmod.org

[4] SAPI. Microsoft Speech API. Microsoft. *http://www.microsoft.com/downloads/details.aspx?FamilyID=5e86ec97-40a7-453f-b0ee-6583171b4530&displaylang=en*

[5] FBX SDK. FBX file format parser. Autodesk. http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=6837478

[6] ARToolKit. Augmented reality library. HIT Lab http://www.hitl.washington.edu/artoolkit/