Technical Aspects of a System for Teaching Aboriginal Languages Using a Game Boy

J. R. Parker
University of Calgary
Digital Media Laboratory
jparker@ucalgary.ca

Ryan Heavy Head
Kainai Studies Department
Mi'kaistoo Community College
Blood Reserve

K. Becker
University of Calgary
Graduate Division of Educational Research

***Abstract*—Aboriginal languages all over the world are threatened with extinction. Aboriginal youth from Canada to New Zealand are not becoming fluent in their language and culture, and the number of fluent speakers is declining severely. The use of computer games in general, and portable platforms in particular, is proposed here as a partial solution to the problem.**

***Index Terms*—audio, computer games, language learning, aboriginal language preservation, audio display.**

1. Introduction

IN 2004, a small group of researchers in the Digital Media Laboratory at the University of Calgary had the idea to create a computer game for aboriginal players in Canada and the US. We saw that there were no significant characters of aboriginal heritage or appearance in current games, although it was known anecdotally that aboriginals played computer games of generally the same sort and in the same amounts as other North Americans.

As our design work progressed, a few interesting ideas developed. First was the thought that a computer game can be used like other multimedia objects on a computer: to record and play back real or synthetic scenes in real or imaginary worlds. A computer game that implemented an aboriginal world according to their actual oral traditions would be not only unique, but could be thought of as an encapsulation of their culture, approximate though it may be. It could be used to preserve the culture and teach it to others, especially the children born into the culture.

One of the most important aspects of any native culture that needs to be taught is the native language itself. Many aboriginal languages are endangered, not just in North America but all over the world. For this reason we decided to focus our attention initially on the language issue in particular.

The University of Calgary has a preexisting relationship with Red Crow College near the town of Standoff in southern Alberta. The college is on a reserve, and the Blackfoot scholars who work there have an interest in the preservation of their language and culture. It is for this reason that the language and culture that we propose to use as a basis for our prototype game project are those of the Blackfoot.

There is no reason why this has to be the only language, though, and if the design is carefully done then the same engine can be used to create language games for many distinct groups. There seems to be no specific design methodology for *multicultural software* of this sort, especially when combined with the artistic and educational aspects that we are encountering, so in a real sense the basic concepts of multicultural software are being devised as we go along.

The game we are building is called ***I'powahsin,*** the Blackfoot word meaning ‘speak’.

2. The Game Design

Although the essential design and programming of the system began in June of 2005, this was done in advance of story details to provide a basic engine for the game. The narrative was missing, and only a shell was in place. The idea underlying the game is that of a quest, a common theme in both Blackfoot stories and in video games. The player will have a main goal and a set of subgoals, but will not know how to achieve the goal at the outset. Instructions will be given to the player in the Blackfoot language at specified points along a predefined route through the game world. These instructions are given in spoken Blackfoot. If the player does not comprehend the instructions, then the next sub-goal in the sequence can’t be achieved. The game will detect these events and prompt the player with

corrections, and will deduct points for lack of comprehension. The game AI will also offer to translate the instructions for a fee (in points or game experience). Success in the task indicates language understanding, and gains points for the player.

So, to make the scheme quite clear: instructions for the player to follow are given in the language to be learned, and a translation can be requested, which costs points. Successfully following the instructions implies language understanding at some level, and game points are awarded. Failing to follow the instructions costs points, and forces the game to accommodate the new situation by either starting over or creating a new set of instructions using the new situation.

The game AI system attempts to determine which specific words and phrases have been used before, and how often. It also keeps track of translations, and attempts to classify words as having been learned by the player or not, and as being difficult for this player or easy. Penalties become greater with each actual translation, and learning difficult words is worth more points than learning hard ones.

Once the story was completely written, the script was created, and all of the spoken sentences needed for the game could be recorded and saved in audio files. Each word in each file is kept in a special internal index; each time a file is played (I.E. each time a sentence is spoken by a game character) all of the words in that sentence have their corresponding usage count incremented. This usage count, or number of times a particular word has been heard by the player, is thus easy to access, and can be used to determine penalties, rewards, and overall performance.

2.1. The Basic Narrative

On June 29, 2005, the principals of this project met at Red Crow College. The goal was to determine the narrative that would be used as a basis for the game play. Although there are many stories from the Blackfoot tradition that could be used, we decided on an age range of 10-14 as a target, and this affected the choice of story. It was decided to create a war story in a traditional fashion based on existing stories of the type. Use of any particular war story would have defined a particular course of action, and would have thus predetermined the play possibilities. Narrative is the enemy of interaction.

Culling a number of themes and strategic details common to classical Blackfoot war tales, an original narrative was designed. The story begins with an animated video. Maanikapi (bachelor) is a young man in love with Iikitsiwaakii (beautiful-woman), the daughter of Omahkínaa (old-man), who is a powerful tribal leader. In hopes of marrying his sweetheart, Maanikapi sends his only horse to Omahkínaa as a gift, along with an emissary who imparts his amorous request. Unfortunately, though, the chief is not impressed with the suitor's offering. Concerned about his daughter's future welfare, Omahkínaa returns the horse to Maanikapi, along with the message that Iikitsiwaakii will not marry a poor man. Given these circumstances, Maanikapi is encouraged by Saahkinaa (young-man), his best friend, to travel southward with him on a horse raid against the Crow. If they are successful, and return with more horses, Maanikapi might then be granted permission to marry. The two agree on this plan, and go to seek the guidance of Naatoyiitapi (balanced-life), a holy man, who makes a sweatlodge for them, and gives them instructions on how to accomplish their goal.

It is at this point where the animated video concludes and the game proper begins. The first stage requires the player, Maanikapi, to visit relatives around their camp in request of a number of supplies and weapons that Naatoyiitapi had instructed him to bring. Each such request will require the player to select language that fits the kinship relation he has with the addressee, and to respond appropriately to any remarks or questions they might make. If approached properly, Maanikapi may receive special charms from some of his relatives, along with knowledge that will help him on his quest. Once all the items have been secured from around their village, Maanikapi and Saahkinaa can embark on their journey.

Moving at night, and camping during the day, the two head south following a prescribed route. All along the way, they face dangers and strategic challenges that Naatoyiitapi had counseled them of. The advise of Naatoyiitapi, as well as any words of wisdom received from Maanikapi's relatives, can be replayed like flashback memories when the partners are in sight of given mnemonic signs. Maanikapi will also be allowed to ask Saahkinaa questions

when in doubt about how to negotiate a particular challenge. Chance encounters with enemy war parties or grizzly bears, the inability to communicate properly with other Blackfoot travellers, failure to secure ample food and water, straying off the charted course, etc., all can lead to disaster for Maanikapi and Saahkinaa. The classic Blackfoot war stories tell of many strategies that one must use while travelling. An encounter with buffalo in the dark of night can startle herds into stampeding, a danger unto itself that may also signal one's location to nearby enemies. In the daytime, magpies, hawks, and waterfowl might give away the player's position, as will an unconcealed fire at night, or smoke during the daytime. Maanikapi will be challenged to read and respond to his environment accurately in order to arrive safely, some nights later, at the Crow encampment.

Once having made it to the enemy village, Maanikapi and Saahkinaa will once more be challenged to use the strategies they've been instructed in. Their goal is to hide-out in the daytime, determine an escape plan, and then stalk toward the encampment on a dark night and steal as many horses as they can without being detected by the enemy. There will be horses at pasture, guarded at all times by a couple of young boys. But the best and most prestigious horses will be staked beside the lodges in camp, with a tether that attaches to their sleeping owners' wrists. Maanikapi can attempt to steal any horse he wants to, but he'd better have carried a charm from his relatives if he hopes to succeed at stealing one of the prized buffalo runners in camp, and he'll have to be careful not to encounter any of the Crow residents who occasionally come and go from their lodges in the night. If they have followed all of their instructions properly, Maanikapi and Saahkinaa will escape undetected. Otherwise, they will only be able to steal so many horses (a number determined by the the points they'd earned previously in the game) before they are found-out and chased by the enemy. Once discovered, Maanikapi will have to evade the enemy and return to a hiding position that he and Saahkinaa will have decided upon prior to their engagements in the camp. Failure to execute the agreed plan will lead to death, and force the player to repeat the horse raid should he chose to try again.

After Maanikapi has successfully escaped from the Crow camp with his stolen horses, a second animated video will play the game's conclusion, showing the partners returning triumphantly to camp with their horses. In the midst of celebration, Maanikapi will once more offer his horses to Omahkínaa. If he has brought back at least four buffalo runners, or eight lesser horses, the offer will be accepted, and Maanikapi will take Iikitsiwaakii as his bride. Otherwise, the gift will be refused, and Omahkínaa will again inform Maanikapi that his daughter is not to marry a poor man, try again.

2.2. Artistic Assets

The long term goal of the project is to use art and music that has been created by aboriginal artists. This will provide the game with the special look and feel that is needed for this new class of game.

Traditional music will be used, as well. Drawings used in the prototype will be based on native renderings, but the final game will use drawing created by native artists trained specially and engaged specifically for this task.

All of the voice clips are recorded in both English and Blackfoot. The recordings are made at the Red Crow site by native speakers, at full speed and at half speed. The half speed versions are used in a 'primer' version of the game, which is otherwise identical to the full speed version.

3. The Game Boy As A Computer

There are a few issues that make this game a challenge to implement, especially on a small device like the Game Boy. The main issues are connected with audio, and with the artificial intelligence system, specifically the analysis of speech, recognizing and rewarding successful learning, and penalizing errors and translations. The Game Boy Advance is *physically* small, but is it small in terms of its abilities as a computer?

The Game Boy Advance has at its heart a 16.7 MHz 16/32 bit ARM7TDMI processor[1]. This is a RISC machine used in many cellular telephones; in fact, over half of all embedded 32 bit CPUs in use today are of this type. This device is as powerful as many desktop computers available in 1993. The display is a 2.9" LCD with a resolution of 240x160 pixels and a colour depth of 15 bits, or 32768 colours.

The graphics processor on the GBA is quite

sophisticated, having three tiled display modes and three bitmap modes, along with 96Kb of dedicated memory. The overall memory capability is 32 Mb without bank switching to the Flash memory card, and a typical Flash game memory card is 256 Mb or less in size.

The most important thing to know about the Game Boy Advance is that specialized hardware activity is all *memory mapped*, so most control activities are accomplished by reading and writing specific memory locations. These locations are often called *registers*, but aren't really.

3.1. Programming the GBA

Although it is possible to conceive of a C compiler that would run on the GBA, this would be impractical for many reasons. The usual situation involves a cross compiler running on a PC creating code for the GBA. Testing is done on the PC through an emulator. Development systems of this sort can be downloaded from many sites on the Internet [7]. The specific system we used was *VisualHAM*, although we had a second system in place to make sure that the behaviour of our software was same on both.

3.2. Playing Sound on the Game Boy

There are very few Game Boy games that include speech as a feature. Why this is can be guessed: the original Game Boy had no digital to analog converter, so older games could not play speech. On the Game Boy Advance (GBA), games implement speech by playing back sound files, and sound files tend to be very large. A Game Boy Flash memory cartridge typically has no more than 256 Megabytes of storage for everything: code, art, music, and other sounds. Sound effects and some music can be implemented by using the build-in sound synthesis features of the Game Boy (or the PC sound card), but voice can't be done that way. Unless voice is essential to the game, it tends to be avoided.

The Game Boy Advance has two 8-bit Digital to Analog Converters (DACs). Each one converts a binary number into a voltage that is then sent to a speaker of headphone. A sequence of binary numbers sent to a speaker quickly enough can reproduce any sound; this is basically how CD and DVD players produce audio. Because there are two DACs, the GBA can play sound in stereo, allowing positional audio, among other things.

The traditional Game Boy sound system has four audio channels, each one being a sound synthesis module. The Sound 1 and Sound 2 channels are each a square wave generator, where channel 1 has a variable duty cycle, frequency sweep and envelope functions, and where the Sound 2 channel has no frequency sweep. Channel 3 acts is a 4-bit DAC that repeatedly plays a pattern of user definable samples. Sound channel 4 produces noise with an envelope function. These are used as a synthesizer to create simple sound effects and cartoon-type music.

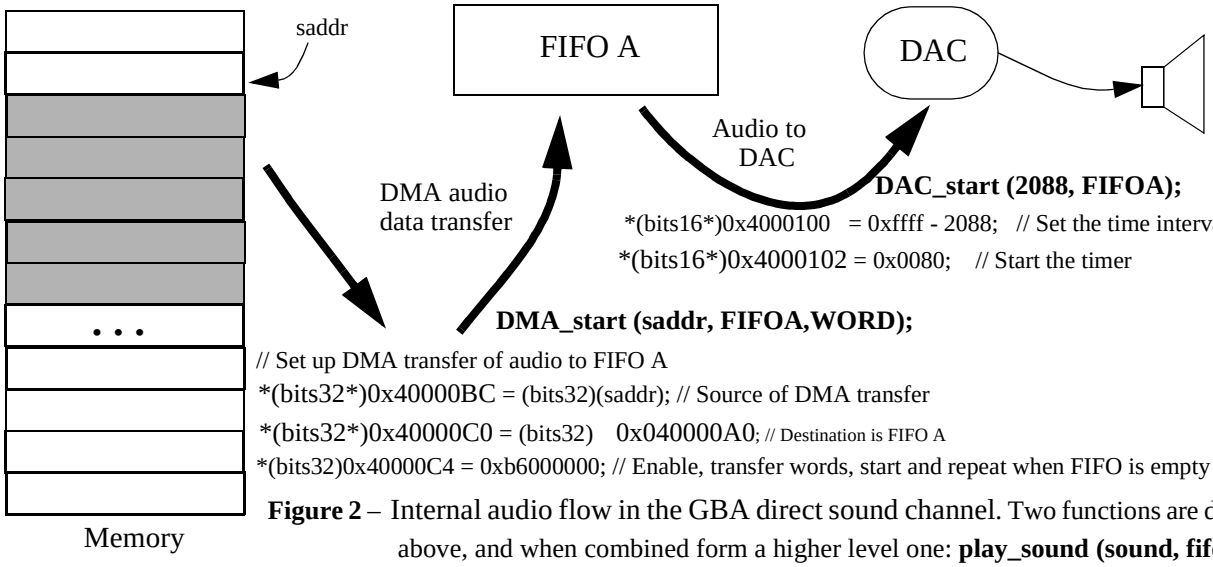

Figure 2 – Internal audio flow in the GBA direct sound channel. Two functions are defined above, and when combined form a higher level one: **play_sound (sound, fifo)**

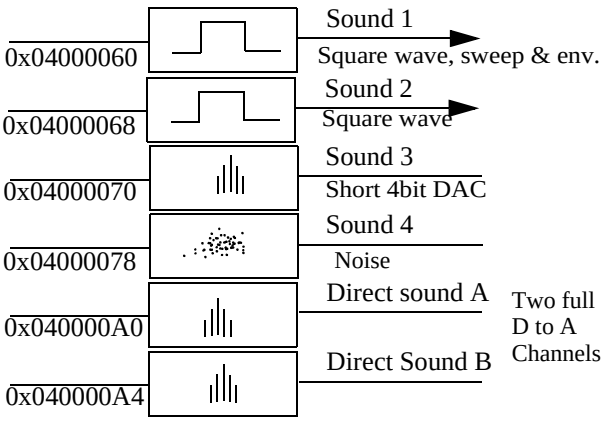

Figure 1 – Game Boy Advance audio channels. Hex number on the left is the base address of the device register.

The DAC channels, only found on the GBA and above, are called the *direct sound* channels by GBA programmers. This use of 'direct sound' is not at all connected to the MicroSoft DirectX facility of the same name. These can play pre-recorded and digitized sounds, like actual music and voice. It is the direct sound that permits the GBA to be used as a language teaching tool.

3.3. A GBA Audio Engine

The idea behind the audio engine that we devised was to create a general, simple interface that could be used in the ***I'powahsin*** project as well as in future games without adding a significant degree of overhead calculation to any game that used it.

Our (necessarily simple) audio engine is based on two components: a pre-processing system (sound pipeline) and an audio API. The pre-processor accepts a WAV format audio file and creates a C-style declaration of an array containing the 8-bit sound samples. A Game Boy has no disk drive, of course, and so sound files must be turned into memory resident data. Our *wavcnvrt* utility does this, and appends array declarations and initiialzations to a specified '.h' file that will be compiled with the game.

The remainder of the engine is a C library that permits the control of the sound channels and passing of data to them. Because the GBA sound system is more primitive than that found on a typical PC, controlling it is more involved. The language project requires speech, which uses only the direct sound channels, but controlling these is a somewhat arcane process, typical of low-level embedded systems. The basic operation of each DAC is to play a single sample when instructed to; thus, if we want to play a sound at a rate of 8000 samples per second, our program must send one sample every 1/8000 seconds (0.125 milliseconds). At 16.7 MHz processor speed this is one sample every 2088 CPU cycles. We need a timer, and a way to transfer data quickly.

The GBA uses a FIFO register to buffer the sound data, which can be loaded using a traditional direct memory access (DMA) scheme. There are two FIFOs, one for each direct sound DAC, and the process of playing a sound from memory is a two-step sequence as shown in Figure 2.

We defined a function named ***DMA_start*** that loads the control registers and begins the process of filling the appropriate FIFO. The FIFO can refilled automatically when it has been emptied by the DAC. We defined another function, ***DAC_start***, that establishes a timer used to move a sound sample from the FIFO to the DAC for playing. In Figure 2, the addresses used for the control and data registers are given in absolute form, as defined by the GBA documentation, and the situation is that described above: an audio data set that has been sampled at 8000 Hz. The timer in the GBA adds one to a 16 bit counter every cycle, and sends a timer signal on overflow. Thus, the register is set to a value of 0xffff-**time**, where **time** is the number of CPU cycles between samples.

The low level engine is responsible for controlling DAC and DMA settings. At a higher level, logical operations such as **play_sound()** are implemented, and there are functions set up for audio mixing of up to eight channels: **link_channel()**, **channel_volume()**, and so on.

Finally, there are functions that are called by the AI system to speak particular phrases. These refer to phrases symbolically, because sometimes there are multiple phrases that mean the same thing. The system selects the specific phrase to be used either based on a random choice, using prior use information, or based on the specific words in the phrase and which have been successfully understood by the player in the past.

4. THE AI SYSTEM - REWARDS AND PENALTIES

The artificial intelligence (AI) module of a computer game is responsible for many key aspects of the game's function, including object collisions,

physics, object management generally, and game play. In the case of **I'powahsin**, the AI system is also responsible for a rather complex scoring system that is based on an educated apprehension of how the language learning goals are being met.

There are two parts to the scoring system: penalties and rewards. The reward system is relatively simple: each phrase is assigned an initial point value, and if a player succeeds at interpreting a phrase, as indicated by successfully following the instructions contained in the phrase, then they are awarded that number of points. Each time the phrase is successfully understood, a decreasing number of points is awarded, but success is always rewarded to some degree. The implementation uses a usage count and score value for each phrase in the game. The value of the reward is $S \times e^{count-1}$ points, where **count** is the number of times that the phrase has been successfully understood and **S** is the initial score assigned to the phrase. Scores are assigned to phrases by native speakers according to how difficult the words and sub-phrases are.

Penalties are assigned differently. Each word in each recorded phrase is assigned a point value, and the total points for a phrase starts out as the sum of the words. Then points are added or removed for combinations; commonly occurring combinations increase the penalty, while rare ones may not, or increase the penalty by a lesser amount. Words that have been previously understood successfully have a double penalty applied. Finally, each phrase increases in penalty value as a function of the number of times that the phrase has been used. If a phrase has been used N times so far and has a basic penalty cost of P points, then the penalty for failure to understand (or cost to translate) is N*P points.

The higher penalty applies to phrases that have been successfully understood in the past. So, if a phrase has been understood 10 times but the player fails in the 11th time, the penalty is 10*P, and the phrase is still treated as one that has been understood 10 times previously. You cannot increase your score by starting the scoring process over again.

The game AI will attempt to determine which words and phrases are difficult for the player, and can create a simple report for the instructor. In more advanced versions of the game, the AI system will test specific hypotheses (E.g. the 'word "xyz" is not understood by the player') when it determines a pattern in understanding through game play.

5. Conclusions

It will be a few years before any reliable assessment can be made concerning the efficacy of the use of the Game Boy to teach native languages It *is* clear that the idea has a feasible implementation at a reasonable cost.

The game in its current form can be downloaded from the ***I'powahsin*** web page at http://www.ucalgary.ca/~jparker/I'powahsin/index.html. The **test.gba** file can be played on any of the publicly available emulators, or it can be written to a flash memory card and played on an actual Game Boy Advance.

Acknowledgments

The authors thank Deifante Walters and Eric Yeung for help with the first versions of the software, and Bailey Parker for some of the initial artwork.

References

[1] ARM Limited ARM7TDMI Technical Reference Manual, Rev. 4, 2001 http://www.arm.com/pdfs/DDI0210B_7TDMI_R4.pdf/m_arm7tdmi.htm

[2] Jonathan S. Harbour, *Programming the Game Boy Advance: The Unofficial Guide*, 2003. http://www.jharbour.com

[3] J. R. Parker and S. Chan, *Sound Synthesis for the Web, Games and Virtual Reality*, *SIGGRAPH 2003*, San Diego, CA. July 28-30, 2003.

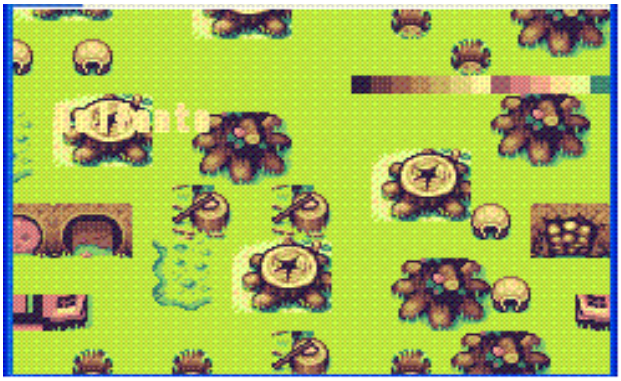

Figure 3 - A sample screen from the ***I'powahsin*** prototype.

[4] J.R. Parker and S. Chan, *OceanQuest: A University Based Serious Games Project*, DiGRA 2005, Vancouver, June 17-19, 2005.
[5] J.R. Parker and K. Becker,
[6] Chris Strickland, Audio *Programming on the GameBoy Advance Part 1*, www.gamedev.net/reference/programming/features/gbasound1/, 1999.
[7] VisualHAM, http://visualham.console-dev.de/, 2005.

Note to reviewers and Futureplay organizers:

We took you at your word, and are submitting this paper in an somewhat incomplete form. It is very new work, and the project is still under way as this is being written. Please consider this submission to be much more than an extended abstract, and somewhat less than a finished paper. We look forward to reviewer's comments.

Playable should be complete in September.

... the authors