KEEPING LEARNERS ENGAGED USING RANDOMIZED PROGRAMMING WITH FEEDBACK

Koji Arizumi 有住幸二, The University of Alabama

ABSTRACT

A good classroom teacher makes many on-the-spot decisions during class, depending on the behaviors of students at each class meeting. Good computer programs used to teach various language skills should do the same. Language activities written to make spot decisions based on user behaviors simulate a good classroom teacher, keeping students engaged with the material. This kind of programming can evaluate a student's level and learning style, and optimize questions suited to it. Randomization of the program alleviates boredom, thus ensuring the students will access the language activities frequently. This paper will present some examples of randomized coding.
Keywords: AI, online, Japanese, simulation, randomization, programming

INTRODUCTION

The majority of online computer language activities consist of a simple interaction between the computer and student through a format consisting of questions generated by the computer and answers input or selected by the learner. Once complete, the student knows all the answers and will not interact with the activity again. A paper quiz with an answer sheet can accomplish the same result as this type of computer program. What students need is a language practice game or quiz that adapts and changes to their ability and learning preferences. This activity should lure the student back again to reinforce what is learned. The activity should progress to higher levels, challenging the student to keep going. Gaming programs that captivate students for hours are difficult to develop, but with the snippets of code presented below, stimulating and adaptable language activities can easily be created. Such activities include a personalized element, much like private tutoring with an experienced teacher's guidance. This teacher would react very quickly to the student's behaviors, knowledge, and non-verbal signals. Computer programs can and should do this as well.

A major weakness of computer programming in foreign language education is the computer's inability to process any sophisticated or subtle input from the students. The usual interaction between computer and learner is: 1) present question, 2) input answer, and 3) tell the student he's right or wrong. There is no interpretation of what the student inputs or how the answer was input. Once the student finishes, it is likely he will not do the game or quiz again, and most likely will forget the material. Computer database systems today, such as Amazon.com, can remember what you buy online and suggest other items you may like, so, why not create language activities that remember student's previous interactions as well? They also draw the customer back by opening with different items each refresh / reload of the page. We need to follow their example.

RANDOMIZATION

One of the strongest programming tools used in gaming products is randomization, which creates the "unpredictability" function. Unpredictability can alleviate boredom when interacting with language learning material. To produce random questions, there are pseudo-random number generators in many computer languages. For example, Math.random() generates a very good simulation in Action Script (C and JavaScript uses a slightly different code). The following code will make one order of numbers change to another order:

```
function narabikae(first, dummy) {
        var first:Array;
        var dummy:Array;
        for (var i = 0; i<first.length; ) {
                flag = false;
                var num:Number = Math.round(Math.random()*(first.length-1));
                for (var j = 0; j<i; j++) {
                        if (first[num] == dummy[j]) {
                                flag = true;
                                break;
                                i--;
                        }
                }
                if (flag == false) {
                        dummy[i] = first[num];
                        i++;
                }
        }
}
```

This code can be applied for changing the order of questions, or the location of certain objects each time the computer presents a question. The following is an example of generating four different questions written in different orders on one screen each time in Action Script:

```
var location:Array = [150, 200, 250, 300]; //y-axis
var new_location:Array = new Array();

narabikae(location, new_location);
        phrase00_txt._y = new_location[0];
        phrase01_txt._y = new_location[1];
        phrase02_txt._y = new_location[2];
        phrase03_txt._y = new_location[3];
```

Thus, the order of any number of questions can be changed each time, or if one applies the concept to location of pictures on the screen for any questions, students can play the game many times for review without boredom.

PERSONALIZATION

It is possible for a computer to store each student's data, such as known vocabulary and grammar structures, and then the language learning program should be able to change its interaction according to each student's past behavior. Feedback can be given by storing student's mistakes during the quiz or game program, and in addition, the computer can slow or speed up its presentation of items, or go back to a more appropriate level based on what the student is doing. Once a computer program can store the student's mistake data, it can analyze the mistaken patterns, and change to the next question/question group accordingly. This can be done using either client-side (via scripting) programming, or online by with various database technology such as MySQL.

If the student makes the same kind of mistakes at certain times, the computer should realize the student is having difficulty comprehending the concepts, and will go back to the easier questions. When the student answers correctly a certain number of times in a short time, the computer should evaluate that he understands the subject and the questions about the same subject should be eliminated, or reduced in number to avoid boring the student.

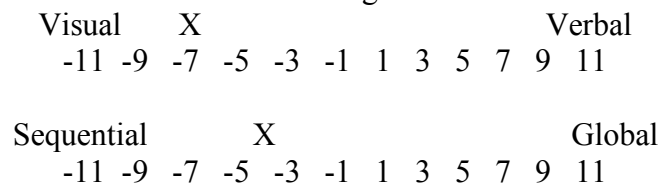
A second personalization possibility is using the computer to evaluate a student's learning style such as visual/auditory or concrete-sequential/global before the questions start, by asking survey-type questions. This can be done through presenting several personal questions such as:

Example Question: When you drive to a certain place:

a. I like to use instructions on how to get there.
b. I like to use a map.

After the computer stores the learning style in number format from the student's answers, it can optimize the questions to create a more comfortable learning environment. For example, the following figure shows a student who has a strong visual learning style and a moderate tendency to prefer concrete-sequential (step by step) learning:

Figure 1



The computer knows the student has a visual learning style, and that his comfortable learning environment is with pictures rather than reading sentences. If a student is of a concrete-sequential learning style, the best way should be step-by step, and very consistent with one subject (topic) learning. The computer should then follow the instruction/questions without variance, moving in a logical pattern from easy to difficult,

or building on previous material in the same way. This is how we can decide the absolute best question presentation pattern for each student's situation:

```
var vv:Number; //Visual-verbal factor
var sg:Number; //sequential-global factor
var original_index:Array;
var original_question:Array;
var decided_question = new Array; //questions chosen
original_index = [4, 8, 8, 3, 8, 8, 6, -3, 2, 0]; //questions learning style index as a number
original_question = ["質問①", "質問②", "質問③", "質問④", "質問⑤", "質問⑥"];
j = 0;
function choose_pattern(vv, sg) {
        for (var i = 0; i<original_index.length; i++) {
                var num:Number = Math.round(vv+sg);
                // some function to decide the index number from learning style
                if (num == original_index[i]) {
                        decided_question[j] = original_question[i];
                        trace(decided_question); //output the question
                        j++;
                }
        }
}
```

CONCLUSION

In conclusion, an effective computer language activity program should behave like a good classroom teacher. It should remember what students know well or where they are weak, and present the appropriate material and feedback to assist the student. It should also take into account each user's learning style and adjust the activity to personalize the material even further.

By using preliminary stored data on each student's level and learning style, combined with mistake analysis, and random number function, the computer program can adapt to the student's learning preference and ability. The program will decide question order and style, and present and choose the most effective questions in the best possible presentation order for each student. The language activity can be repeated many times, and not only does it alleviate boredom, but adapts as the student learns the material.

REFERENCE

Oxford, Rebecca (1994). *Styles Analysis Survey*
http://www.as.ua.edu/nihongo/sas/survey.html
Sanders, William B. (2004). *Macromedia Flash MX Professional 2004 Kick Start*, IN: Sams Publishing