Development of Sample Code Stocks for AR Applications in Language Learning

Hajime Mochizuki

Abstract—**Augmented Reality (AR) applications provide users with an experience in which the real world is enhanced with projected virtual objects that are perceived by the users as being present in the real world. Learning applications can be made more interactive and engaging through AR technology. Along these lines, open-source codes with sample applications for AR are available for download on the Internet. However, if users needed to make their own applications, they would require to perform their own programming to suit their specific needs. Slightly modifying a program is not very difficult for a user who is proficient in programming. However, not all users are good programmers. In this paper, we develop program sources as sample code stocks for inexperienced programmers. We describe the specific details of slight modifications in which a sample program is expanded to have high utility value for language learning. We also show three instances of AR applications wherein our modified sample programs and multiple 3D objects are used by inexperienced programmers.**

Index Terms—**Augmented reality (AR), language learning, edutainment, ARToolKit.**

I. Introduction

Augmented reality (AR) technology overlays computer-generated objects such as 2D and 3D models, texts, pictures, and video images with physical objects, such as specific markers, on images captured successively by a video camera in real time [1]. Composite images with AR objects are viewed through many kinds of computer devices such as a head mounted display (HMD), a projector, or a computer screen. AR applications afford a novel experience in which the real world is enhanced through projected virtual objects that are perceived by users as being present in the real world. Learning applications can be made more interactive and engaging through AR technology [2].

The 2011 Horizon Reports from the Higher Ed Edition of the New Media Consortium stated that "one of the most promising aspects of augmented reality is that it can be used for visual and highly interactive forms of learning, allowing the overlay of data onto the real world as easily as it simulates dynamic processes" [3].

A novel experience within an augmented world viewed by a user through a lens can make learning applications interesting, pleasing, and stimulating. AR inspires not only application users but also application developers. For example, a language teacher or a person who is interested in a language learning system might be inspired to prepare attractive teaching materials using AR; a simple example of the latter would be a word card application using 3D objects that are overlaid with entry words.

Manuscript received June 3, 2015; revised August 11, 2015.

Hajime Mochizuki is with the Institute of Global Studies, Tokyo University of Foreign Studies, 3-11-1 Asahi, Fuchu, Tokyo, 183-8534, Japan (e-mail: motizuki@tufs.ac.jp).

Fig. 1 and Fig. 2 show a simple example of this application that uses ARToolKit, which is a major open-source framework for easily developing AR applications [4], [5]. Fig. 1 shows a piece of paper with the word "BUS." The piece of paper is used as an AR marker for ARToolKit. Fig. 2 shows the same piece of paper, overlaid with a 3D picture of a bus. The 3D bus was made using Metasequoia, a shareware 3D modeling application. The simple example in Fig. 2 demonstrates one marker and one 3D object. To make the same application, only ARToolKit and the GLMetaseq library need to be set up, and another open-source library that uses ARToolKit's MQO data model format, to prepare the 3D model of a bus and one AR marker [6]-[8]. The sample program can be executed for a single 3D object without compiling source code, by replacing the sample 3D model data and marker with the target data.

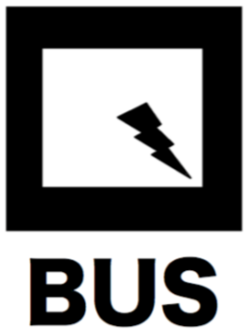

Fig. 1. Sample AR marker for the word "BUS."

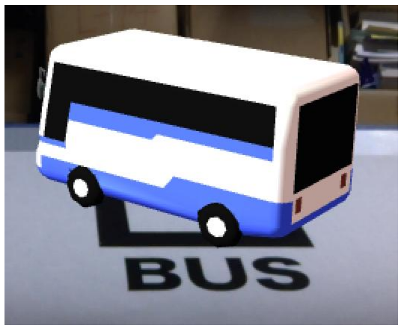

Fig. 2. 3D object on the marker.

However, this implementation works with only one word, and is not sufficient for our simple word card application. If we truly desire a useful word card application, the program has to be written such that it can store information for multiple words. Not all language teachers are good programmers, even though a modest design change for multiple objects may seem simple enough for experienced programmers knowledgeable in the C or C++ programming languages. Implementation of non-programmers' creative ideas can be limited by a lack of programming skills.

We think the problems presented by a lack of programming skills can be solved by providing appropriate sample program sources that meet the needs of inexperienced programmers' ideas.

Our previous work focused on slight modification to supplement programs. We reported the effectiveness of modified code to supplement programs in the development of AR applications [9]. In this paper, we develop such program sources as sample code stocks. We describe the specific details of three modifications in which sample programs are expanded to have high utility value for language learning. We also show three instances of AR applications that use our modified sample programs and multiple 3D objects.

II. Slight Modifications of Sample Programs

Writing programs is difficult for a person who is not proficient in computer programming. The spread of the Internet has led to an abundance of opportunities for discovering attractive software. Furthermore, open source programs are increasingly available for download via the Internet, and usually include both sample source code and executable programs in the packages. The simplest and quickest way to make a new application program is to modify the original application's sample source code.

However, we tend to forget that mastering a programming language requires considerable effort. Writing application programs requires fundamental knowledge and basic skill in computers and programming languages. While most sample programs for open-source applications are easy to execute, thus far they are used as a trial; whenever an unskilled person attempts to customize an application system he/she tends to encounter a problem with programming.

We aim to help them by providing useful modified sample source codes. In this section, we describe three valuable modifications of the sample code of ARToolKit and GLMetaseq after pointing out the main part of an application's original sample source code.

A. Basic Knowledge to Modify ARToolKit Sample Programs

In the case of ARToolKit and GLMetaseq, we have to understand at least the following points about the application's original program source code.

1) A filename of a single marker data is assigned to a char type variable "*patt_name*" in the sample source code of ARToolKit.
2) A filename of a single MQO format 3D data is assigned to the char type variable "*mqo_name*."
3) A single marker ID is assigned to the integer type variable "*patt_id*" in the source code of ARToolKit.
4) To load a marker pattern file the function "*arLoadPatt()*" is called and a return value for the marker ID is assigned to the variable *patt_id* by "*patt_id = arLoadPatt(patt_name)*." Here, the first argument *patt_name* is a filename of marker data.
5) GLMetaseq is used for MQO-formatted 3D data. Here, the GLMetaseq is a library to display a 3D object of MQO format by using OpenGL, a major open-source graphic library.
6) A single MQO model data is assigned to the MQO_MODEL type variable "*model*" in the source code.
7) To load MQO model data the function "*mqoCreateModel()*" is called. Its return value, MQO_MODEL type data, is assigned to the variable model by "*model = mqoCreateModel(mqo_name, 1.0)*." Here, the first argument is a filename of MQO model data, and the second argument means a magnification.
8) The function *mqoCallModel()* is used to draw a MQO 3D object.

B. Modification for Displaying Multiple Static 3D Models

Under the above conditions, if we want to make an application such as noun cards with AR technology, we need to modify the original sample code as per the following points.

1) A filename of a single marker data is assigned to a char type variable "*patt_name*" in the sample source code of ARToolKit.
2) The variable *patt_id* is replaced by the array *patt_ids[MARKER_NUM]* in order to register multiple marker IDs. Here, "*MARKER_NUM*" means the number of marker IDs.
3) The function *arLoadPatt()* is called *MARKER_NUM* times to assign marker IDs to the array *patt_ids[]*. The example modified code is

```
for(i=0; i<MARKER_NUM; i++) {
        sprintf(patt_name, "Data/patt.%d", i+1);
        patt_ids[i]=arLoadPatt(patt_name);
       }.
```

Here, the temporary variable *patt_name* is used to format and register the filename of marker data. We use a naming rule that the string "*patt.*" is followed by a sequential number such as "*patt.1*" or "*patt.2*" for convenience.

1) The variable *model* is replaced by the array "*model[MQO_MODEL_NUM]"* in order to register multiple MQO model data. Here, *MQO_MODEL_NUM* is the number of MQO model data.
2) The function "*mqoCreateModel()*" is called *MQO_MODEL_NUM* times to load multiple MQO models. The example modified code is

```
for(i=0; i<MQO_MODEL_NUM; i++) {
   sprintf(model_name,"Data/noun0%d.mqo", i);
   model[i]=mqoCreateModel(model_name, 1.0);
 }.
```

Here, the temporary variable *model_name* is used to format and register the filename of MQO model data. We use a naming rule that the string "*noun*" is followed by a sequential two-digit number and string "*.mqo"* such as "*noun01.mqo*" and "*noun02.mqo*" for convenience.

Through these modifications, multiple static 3D objects can be displayed simultaneously. A user can use his/her original MQO format files created by the Metasequoia or other 3D modeling applications.

We call this modified sample code "MS3D" in this paper.

C. Modification for Displaying Multiple Animated 3D

Models

Furthermore, if we expect our word cards application to make use of verb entries, the code has to be modified to display multiple animated 3D objects. To create these applications, we need to re-modify the above code in the following manner:

1) *MQO_MODEL* type array "*model[]*" is replaced by *MQO_SEQUENCE* type array "*model_seq[]*", for registering multiple sequential MQO models for each 3D object.
2) The function *mqoCreateModel()* is replaced by the function "*mqoCreateSequence()*", in order to display multiple MQO models sequentially as an animation.
3) The function *mqoCreateSequence()* is called as follows:

```
for (i=0; i<MQO_MODEL_NUM; i++) {
   model_seq[i]  =
       mqoCreateSequence(
   mqo_seq_names[i], frame_num[i], 1.0);
 }.
```

Here, "*mqo_seq_names[]*" is a char type array that stores filenames for MQO model data. The "*frame_num[]*" is an integer type array that stores the number of MQO model frames.

1) To draw MQO 3D objects sequentially the function *mqoCallSequence()* is called instead of the *mqoCallModel()*.

With these modifications, multiple animated 3D objects can be displayed simultaneously. To create an animated MQO format model, any 3D animation software such as RokDeBone2 can be used. The significant requirement for the animation software is a function to generate sequential MQO format files from a single 3D animated object.

We call this modified sample code "MA3D" in this paper.

D. Modification for Switching Multiple Static 3D Models According to Pressed Keys

The previous two modifications aimed at displaying a single 3D model or a series of a single animated 3D model on each AR marker. However, we can assume that there can also be an application where each marker is shared by multiple different 3D objects. Our third modification is for a programmer who requires a function that switches 3D models on each maker according to pressed keys. In this case, we are required to modify the original sample code according to the following points.

1) Similar to the case of noun cards, the variable *patt_id* is replaced by the array *patt_ids[MARKER_NUM]* in order to register multiple marker IDs. Here, "*MARKER_NUM*" means the number of marker IDs.
2) The variable model is replaced by the two dimensional array "*model[MARKER_NUM][KEY_NUM]*" in order to register multiple MQO model data. Here, *MARKER_NUM* is the number of markers and *KEY_NUM* is the number of binding keys.
3) The function *mqoCreateModel()* is called *MARKER_NUM* times *KEY_NUM* times to load multiple MQO models. The example modified code is

```
for(i=0; i<MARKER_NUM; i++) {
   for (j=0; j<KEY_NUM; j++) {
     sprintf(model_name,
         "Data/mqo0%02d%02d.mqo", i, j);
     model[i][j]=
           mqoCreateModel(model_name, 1.0);
   }
 }.
```

Here, the temporary variable *model_name* is used to format and register the filename of MQO model data. We use a naming rule that the string "*mqo*" is followed by a sequential four-digit number and string "*.mqo*" such as "*mqo0101.mqo*" and "*mqo0102.mqo*" for convenience.

1) The function "*KeyEvent()*" is defined for key binding.

With these modifications, 3D image displayed on each marker can be switched according to pressed keys.

We call this modified sample code "SMS3D" in this paper.

III. AR Applications for Language Learning

Ideally, computer programming education should be made more easily available in schools, regardless of a student's majors. However, in many universities, with the exception of science and engineering courses, the number of classes for information education is often kept to a minimum because of a shortage of facilities and teachers, and additionally due to limitations in the curriculum.

As mentioned above, there are users who do not bring their ideas to fruition because of a lack of opportunity to acquire programming skills. One way to avoid such regrettable situations is to recruit the aid of an experienced programmer. If there are appropriate sample programs, inexperienced programmers need not give up on their ideas. We believe that there can be higher opportunities for success if there are more sample code stocks supplied as reusable materials.

The remainder of this paper describes three instances of AR applications in language learning using the modified sample programs stated in section II. All AR applications were created by the author who is a university teacher and inexperienced programmers who are university students. The first two applications use multiple AR markers tied to noun words and verb words. The first two applications use multiple AR markers tied to noun words and verb words. Each card with an AR marker is overlayed with a static or animated 3D model that expresses a noun or verb word. The third application can switch overlayed 3D models for the same marker by typing keys. Therefore, through the third application, we are able to learn multiple languages using single set of AR markers.

A. AR Noun Cards

AR noun cards are very simple. The requirements for this application are to prepare AR markers and 3D models, according to the number of entry words.

We use the sample code DMS3D for this application. Except for the programming, the most difficult task for novice users is to draw 3D models. We use Metasequoia LE for drawing 3D models because it is relatively easy to use. The MQO-formatted models can be rendered with the ARToolKit, when expanded by the GLMetaseq library.

The inexperienced programmers who are university students completed more than ten 3D models in one month, including the time for learning how to use Metasequoia. Fig. 3

and Fig. 4 show examples of our noun cards "train" and "bicycle," respectively.

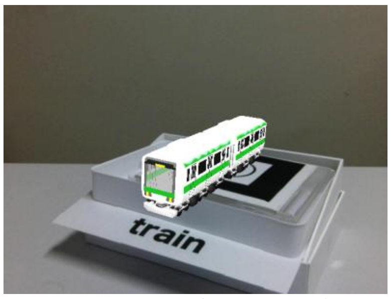

Fig. 3. 3D object of the word "train."

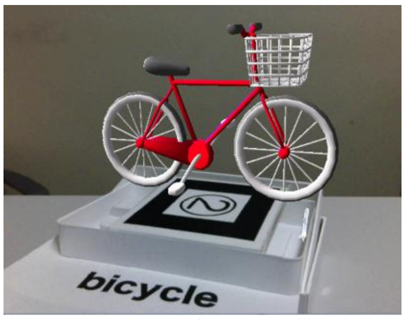

Fig. 4. 3D object of the word "bicycle."

One of the attractive features of three-dimensional expression is the flexibility to view a model from any angle. A user can watch the reverse side of a model by turning the angle of the AR marker to reverse.

As shown in above figures, the quality of our 3D models is very good. However, the quality of 3D model is not very important or essential for AR technology. Even if the 3D model is unrefined, AR applications provide a new experience in which the real world is enhanced through projected virtual objects that can be perceived by users in the real world. AR technology can lend a deep impact and emotion to a user's learning experience.

B. AR Verb Cards

Our verb cards are also simple. The requirement for this application is to prepare AR markers and 3D animated models, according to the number of entry words.

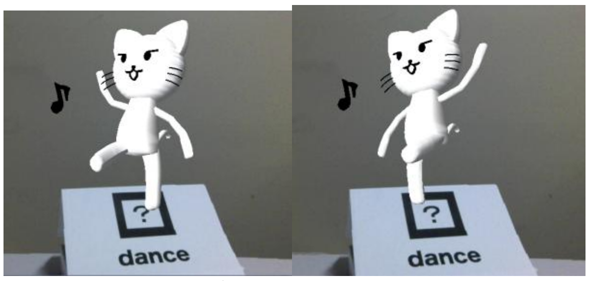

Fig. 5. Animated 3D object of the word "dance." The character in the Figure is the original character of our laboratory designed by Ms. Hitomi Yasui. The character is a cat and his name is "Niko."

We use the sample code DMA3D for this application. As in the noun cards, the tasks include programming, drawing of 3D models, and creation of animations. We use RokDeBone2, which uses skeletal animation techniques to create animation data from MQO 3D model data [10]. The notion of skeletal animation is difficult to comprehend.

Our students learned how to use the skeletal animation software by reading the tutorial pages on the Internet. ARToolKit with the GLMetaseq library can produce 3D animation, by rendering multiple frames of MQO data sequentially. RokDeBone2 can generate sequential MQO data from a single sequence of animation data. Our students completed more than ten 3D animated models, including trial models. Fig. 5 and Fig. 6 show the examples of our verb cards "dance" and "wake up," respectively.

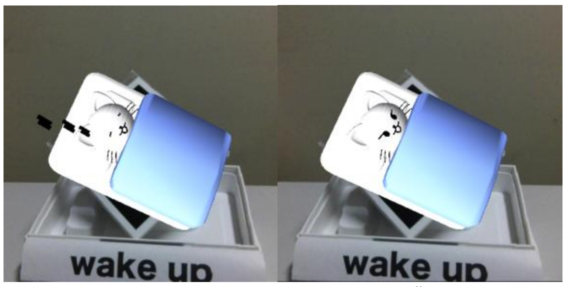

Fig. 6. Animated 3D object of the word "wake up."

3D animation models are more attractive than simple 3D models. We believe that our 3D verb cards can provide a delightful experience for the users. Furthermore, communicating the meaning of verbs is made clear through the use of animations.

C. AR 3D Letter String Labeling in Multiple Languages

The previous two applications display fine quality 3D models that are drawn using 3D modeling software. However, not all programmers are good 3D model creators even if a modest modeling software is relatively easy to use. Therefore, we use more simple 3D objects instead of fine grade 3D objects in our third application. Specifically, we create letter string 3D objects by using StringMQObject Builder, which is a freeware.

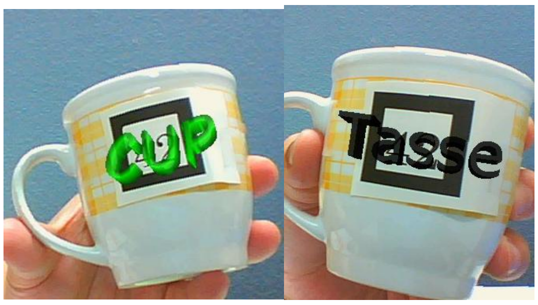

Fig. 8. 3D letter strings that were displayed on the same marker pasted on the real cup. The upper side picture is in English and the lower side is in French.'

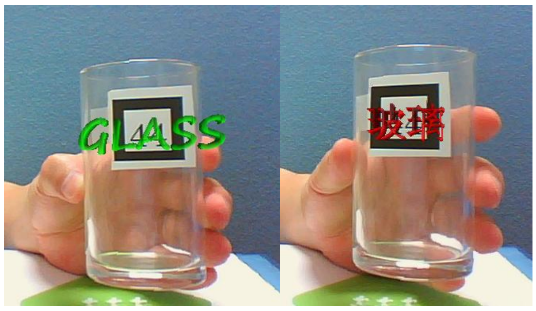

Fig. 9. 3D letter strings that were displayed on the same marker pasted on the real glass. The upper side picture is in English and the lower side is in Chinese.

StringMQObject Builder is very easy to use because a user can make a letter string 3D object from a typed string by performing a few operations on the GUI. The requirements for this application are to create 3D "letter string" objects and to prepare real objects in which markers are pasted. For example, we prepare a real coffee cup and a marker, and we create 3D letter string objects from multiple languages such as "cup" (in English), "koppu"(in Japanese), "tasse" (in French). Fig. 8 and Fig. 9 show the examples of our 3D letter string label application.

The letter string on the upper side in Fig. 8 represents the English name of the real object "cup" and the letter string on the lower side is the French name of the same real object "Tasse." The letter string on the lower side in Fig. 9 represents the Chinese name of the real object "glass." Users can switch the showing language by pressing binded keys. In our application, we register the keys E, J, F, C, R, and D for English, Japanese, French, Chinese, Russian, and German, respectively. To make this application, we use the sample code SMS3D.

The idea that 3D letter string is overlayed with an existing physical object is suitable for AR application. We think this application is excellent in language learning application because overlayed languages are easy to change by only pressing keys.

IV. Conclusion

In this paper, we developed program sources as sample code stocks. We described the specific details of a slight modification in which a sample program is expanded to have a high utility value for language learning. We also showed three instances of AR applications that use our modified sample programs and multiple 3D objects.

Both AR noun cards and AR verb cards made use of English words in this paper. We can also use other languages by changing the words written on the maker papers. Even though writing program source code was difficult for our students who are inexperienced programmers, they demonstrated talent in creating 3D models.

We think the problems presented by a lack of programming skills can be solved by providing appropriate sample program sources that meet the needs of students' ideas. This paper argued for the effectiveness of original sample programs, slightly modified to adjust to our needs for treating multiple models simultaneously. We have shown that by making small changes to an existing program, we can obtain high utility value. As future work, to improve the utility values of our sample code stocks for AR applications, we intend to continue the development in order to increase reusable sample codes.

Acknowledgment

The author would like to thank Ms. Chihiro Yamashita, Ms. Megumi Ito and Ms. Hitomi Yasui for providing attractive 3D objects.

References

[1] W. F. Krevelen and R. Poelman, "A survey of augmented reality technologies, applications and limitations," *The International Journal of Virtual Reality*, vol. 9, no. 2, pp. 1-20, 2010.

[2] S. Yuen, G. Yaoyuneyong, and E. Johnson, "Augmented reality: An overview and five directions for AR in education," *Journal of Educational Technology Development and Exchange*, vol. 4, no. 1, pp. 119-140, 2011.

[3] NMC, *2011 New Media Consortium*, Horizontal Report, pp. 16-19, 2011.

[4] M. Billinghurst, A. Cheok, S. Prince, and H. Kato, "Real world teleconferencing," *IEEE Computer Graphics and Applications*, vol. 22, no. 6, pp. 11-13, Nov/Dec., 2002.

[5] ARToolKit Homepage. (2002). [Online]. Available: http://www.hitl.washington.edu/artoolkit/

[6] S. Hashimoto. (2007). Engineering Navi Homepage. [Online]. Available: http://kougaku-navi.net/ARToolKit/

[7] S. Hashimoto, "ARToolKit: Introduction to programming in augmented reality," *Ascii Media Works*, 2008.

[8] Tetraface Inc. (Feb. 2014). Metasequoia file format specification. [Online]. Available: http://www.metaseq.net/en/format.html

[9] H. Mochizuki, "The Effectiveness of Slight Modification to Supplement Programs in the Development of AR Applications," in *Proc. World Conference on e-Learning in Corporate, Government, Healthcare, and Higher Education*, vol. 2014, no. 1, pp. 1390-1396, 2014.

[10] R. Mukundan and S. Animation, *In Advanced Methods in Computer Graphics*, Springer, pp. 53-76, 2012.

Hajime Mochizuki was born in Japan. He obtained a PhD in information science from Japan Advanced Institute of Science and Technology in Ishikawa, Japan in 1999. He is currently an associate professor at the Institute of Global Studies, Tokyo University of Foreign Studies. His research interests include natural language processing and language learning system. Dr. Mochizuki is a member of the Information Processing Society of Japan (IPSJ), the Association for Natural Language Processing in Japan, Japanese Society for Information and Systems in Education (JSISE), and Association for the Advancement of Computing in Education (AACE).