IPT & EGVE Workshop (2005)
R. Blach, E. Kjems (Editors)

Language Learning in Virtual Environments: 'Bobo and Apples'

H. Holmen and F. Nielsen

Aalborg University Esbjerg, Department for Software and Media Technology

Abstract
'Bobo and Apples' is one of the prototype games within SAME4KIDS (Speech-based,Animated, Multilingual,Educational games for Kids, http://same4kids.sourceforge.net/), a multi-language and multi-purpose games project for young kids of age 3-5 years. The main goal of SAME4KIDS is to expose young learners to multi-module games in various available platforms. In the prototype of 'Bobo and Apples', the game is designed to teach multiple languages and simple math within a frame of virtual environment, using mainly visual images, animation and sound. In this paper we introduce the main design concept and architecture for the prototype, as well as the envisioned VR conversion of the game, based on the Animarium *system.*

1. Introduction

International personal mobility, due to global economy, political situation and education, is common in present time. As a consequence, increasingly many families are left in a situation that requires multiple languages (e.g. home: language 1, work: language 2, nation of residence: language 3, etc). Traditional language educational tools are largely for one language at a time, while in reality multiple languages are often required simultaneously. Although it is known that early exposition to languages is crucial for learning (Tok01), many educational tools do not accommodate the needs of poly-lingual families with toddlers. The SAME4KIDS project's motivation is to find a new methodology for such families, centered on a pictorial translation component. (Figure 1)

Since the project is planned to run both on PCs and on the web, portability is one of our central requirements. Moreover, later in our research, we also plan to evaluate the possibilities offered by mobile hardware platforms, such as mobile phones or PDAs. This is why we decided to develop most SAME4KIDS games in Java and XML; SVG will also be used in the client-side of some of the games and to define and print paper tangibles. Finally, voice recognition and real-time audio manipulation require some third-party, platform specific code, in which case libraries in C/C++ are used (and integrated with Java code as native methods - JNI).

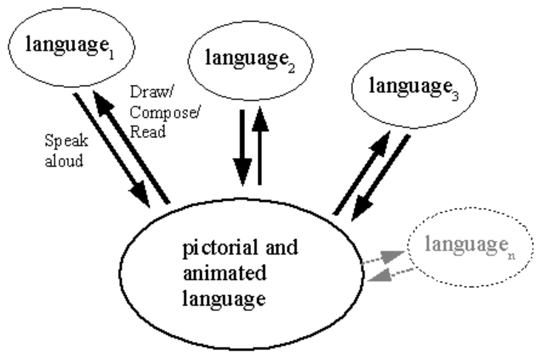

Figure 1*: Conceptual model of the pictorial language*

Meanwhile, 3D VE games such as 'Bobo and Apples', are developed in a more complicated and dedicated programming system, *Animarium*, with intensions to make the conversion between different systems easier. For the SAME4KIDS purpose itself, 'Bobo and Apples' in confined systems such as CAVE might not be a realistic means for many target users at the moment. However, with generated interest in VR and AR, particularly using mobile phone (with cameras), it is possible for researchers to envision the possible future. This paper is discussing an educational game project design in VE and its implementing tool.

As mentioned before, the SAME4KIDS project consists of many small games, specifically designed to experience multiple languages simultaneously. Overall, the project is developing the following technical areas:

- Visual-language translator
- Prototype implementing tools
- Speech recognition

2. Design Concept of 'Bobo and Apples'

As a SAME4KIDS game, 'Bobo and Apples' has the following general requirements to meet:

- Multi-language
- Multi-purpose
- Kids-friendly HCI

The player will be controlling Bobo, the host character. Instructions will be given in speech, as well as written words on the screen. There will be 5 different levels focusing on different aspects of language (figure 2).

Level 1	Level 2	Level 3	Level 4	Level 5
Names	+ size	+ colour	+numbers	All mixed
Apple Banana Strawberry	Big Small Giant Tiny	Red Green Yellow Etc.	1 to 10	Simple math

Figure 2: *Game levels and learning objectives*

Game instructions will be given both as speech and written words. Here is a demonstrative game play scenario in the first level:

1. Open the game; meet Bobo. (Sound: 'Here is Bobo')
2. Controlling instruction. ('Use the mouse to move Bobo around, and click the mouse button to pick up an object')
3. Game begins. Bobo walks around in a small 3D environment with some objects scattered around. ('Bobo wants an apple. Go to the apple.')
4. Pick up the apple. The apple will get legs and follow Bobo. A reward point (an apple in a basket) is given, and when all 5 points are taken, the next level opens.

'Bobo and Apples' will mix multiple languages automatically, simultaneously. For example, instructions to get fruits (step 3) will be given in a randomly chosen language. The player can not choose language within the game. The base-language the game starts with is determined by the log-in process.

In each higher level, more language attributes are added, and repetition of simple words, visual representation, and pronunciation are important methods for learning throughout the project. For example, in level 4, one can still hear the instruction 'Bobo wants an apple', but also 'Bobo wants two, big, green apples'. In level 5, the game can present confusing situations. For example, one hears the instruction of 'Bobo wants one, big, red apple', while the visuals are one big green apple and one tiny red apple. Should you choose the red apple, even though it is not big?

For the prototype research, the project is working with two Danish kindergartens. The two groups are very different, both in goals and computer access. Group A is an average Danish kindergarten where 99% of the kids are Danish. Caretakers of Group A think the tool may help those kids who need to improve their native language (Danish), as well as inspire a few for the second language training.

Group B is largely kids from refugee families from African and Arab nations, who need to learn Danish as soon as possible. Teachers in Group B hope that the tool may help in learning Danish (second language for the kids), and are largely interested in the visual-language translator function of the project. Within this group environment, it is often the case that neither parents and kids nor the teachers share a common language to communicate in a basic level. It might be necessary to customize the game with Arabic in 'Bobo and Apples' for Group B.

Regarding HCI, one important factor with both groups of kids is familiarity with computers, and the use of input devices. In the future, the project with speech recognition would solve many control device problems. The prototype is using mouse and a developed easy-to-use login system with unique id, represented by the kids' own pictures. The login id information consists of the following:

- Country of residence (Base language)
- Each parent's language
- Parent's common language (e.g. in a multi racial family, a Danish father and French mother may communicate with each other in English)
- Preferred 1st and 2nd languages

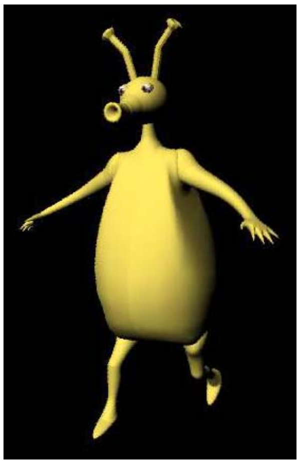

Figure 3: *Bobo*

Animation of Bobo and characters are made in 3D animation software Maya, and the shape and acting style are meant to imitate a small child. Bobo is a rather typical representation of a child (figure 3) in the sense of very short, tiny steps, a funny balance, and bottom heavy design. These are the same ideas as used in famous kids-friendly characters like Teletubbies and Big Bird. The other characters are neutral with no expressive body parts other than legs (figure 4).

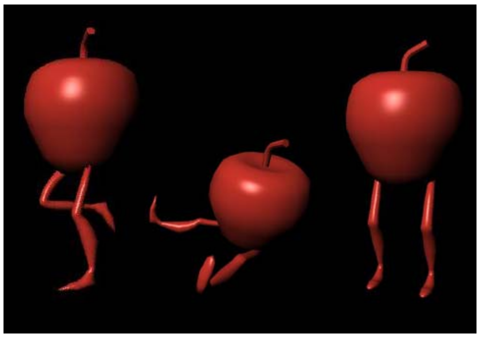

Figure 4: *Character design for the apples.*

3. Implementation in *Animarium*

Animarium is a prototypical virtual Medialogy laboratory. The system was conceived originally as a teaching tool, for a computer game construction course held at Esbjerg campus for bachelor level students. In conjunction with the course, students were to design a computer game and implement a prototype showing key gameplay, as a semester project. When working with computer game-like systems in academia, one quickly stumbles on the problem of available tools. While a large number of software packages / game engines are available, the general tendency of current-crop systems is that they require a large amount of training and very good technical abilities on the part of the users, before actual gameplay can be implemented. In this situation we were faced with the prospect of devoting a large part of the course to teaching the specifics of such a tool, which is something we would rather avoid, instead devoting the course to the more important principles of game design and development. Hence we quickly implemented the first prototype of *Animarium*, using a series of open-source libraries, the chief one being the programming language Python (see `http://www.python.org`).

For this project, the environment was used to create the multi-language learning game "Bobo and apples". IN particular, we will describe how the animated "Apple" objects that grow legs and follow the main character around when touched. It was thought that giving the objects a life of their own would stimulate the children playing the game further than just walking up to inanimate objects.

As mentioned before, the legged apple was modelled in Maya as a skinned character with an animated skeleton. Subsequently, a number of animation "snippets" was created for it (see fig. 5, below).

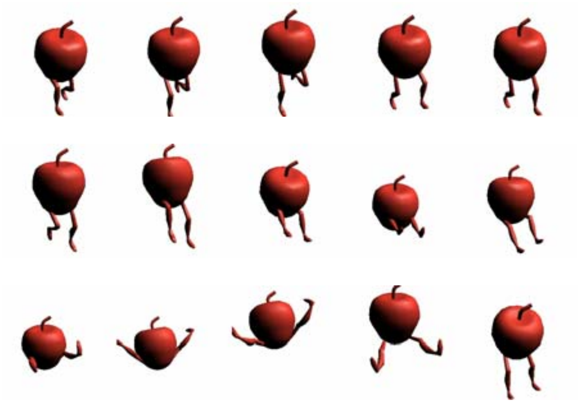

Figure 5: Example of three animation 'snippets' for the apple – 'walk', 'fall' and 'getup'

In addition to the animation, a basic behaviour pattern was construed with the object of installing some life and fun in the apples. This took the form of a list of predefined behaviours:

- **Sleeping** – before being approached by the player, the apples are inert, giving surprise of growing legs and becoming alive when touched
- **Waiting** – when alerted and being sufficiently close to the player, the apple simply stands in its place using an "idle" animation cycle
- **Following** – whenever the distance between the apple and the player becomes too great, the apple tries to bring itself closer by walking. The algorithm is exceedingly simple, just walk forward whilst always turning in the direction of the player
- **Falling** – to underline the comical nature of the apples-with-legs, it was decided to animate the apples bumping into an obstacle, i.e. falling on their behind and getting up again. In order to avoid the obstacle again, the apple simply turns 135º to the left and starts walking again
- **Searching** – if the line of sight between the player and the apple is broken, it immediately forgets where the player was, and starts walking aimlessly around, in the hope of accidentally re-gaining visual contact

Subsequently, a script for the objects was designed, in form of a state machine. Each behaviour on the list was turned into a state in the script, and a number of transitions with adjoining conditions were added. The design is illustrated in fig. 6, below:

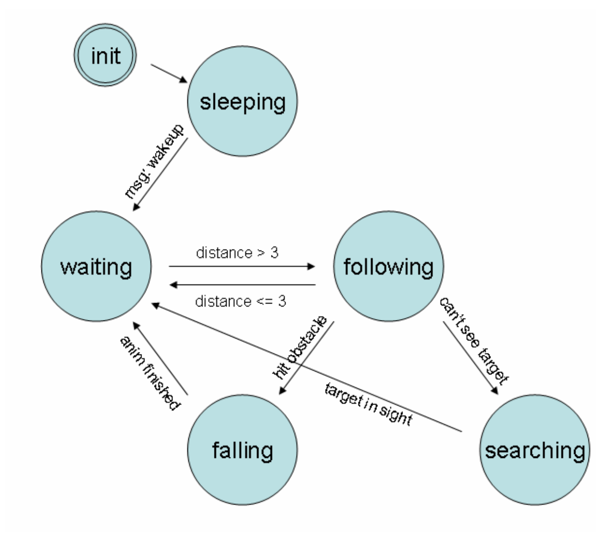

Figure 6: *State-transition diagram for the autonomous "Apple" object. Arrows mark transitions along with the condition that cause them.*

4. Architecture of *Animarium*

The core activity of implementing gameplay is programming (or "scripting") of behaviour in autonomous actors. Based on earlier experience from commercial game development, we wanted to teach the students the concept of building state machines in a concurrent programming environment. One reason for choosing Python as both implementation and scripting language for the system is the availability of a language extension known as Stackless Python (see `http://www.stackless.com`), which extends the language semantics to allow ultra-lightweight microthreads ("tasklets") to execute simultaneously in a voluntary time-sharing scheme. On this foundation a very light scripting framework was built, which allows objects in *Animarium*'s 3D environment to be controlled by concurrent state-machine scripts, written in Python. These scripts would control the appearance, movement etc. of the objects and communicate with one another using synchronous message passing.

4.1 State machines

One reason for choosing state machines as the main way to structure scripts is the suitability of this tool for interactive animation. Much of the animation in the prototype is realized using pre-recorded animation snippets, both loops and one-shot animations, which are played in series. In many cases, a single animation snippet corresponds exactly to one state in the script, so the object can be animated using a simple mapping of state-to-animation snippet. As an example of this, consider the behaviour of the "fruit" objects in the prototype, and the two behaviour patterns of a) wandering aimlessly around in search of Bobo when they cannot see him, and b) bumping into a wall which happens to be in the way. Both have a distinct animation ("walk" (a) and "fall+getup" (b)) as well as a distinct state ("searching" and "falling") associated with them. The visualization of these two behaviours is achieved with a simple mapping of the states (in the script), and corresponding animation 'snippets' (see earlier fig. 5).

4.2 Concurrency and the Actor model

The current implementation of *Animarium* relies on segregated concurrent processes communicating via message passing. It was thought that teaching this tool to the students would ease the difficult task of creating a working interactive system with a high degree of parallelism, which a computer game with autonomous behaviour inevitably is.

This paradigm of computation originated in research done at MIT's Artificial Intelligence Laboratory starting in the 1970's. The so-called Actor model developed in main by Carl Hewitt [Hew76], and the term *concurrent* programming defines the terminology, and there was a later formalization in the form of the *pi-calculus* [Mil99]. One should consider the paradigm as an alternative to multiprogramming, the currently more common approach to parallelism involving multiple parallel processes ("threads") communicating with each other using shared memory.

It is our belief that this programming paradigm will reveal itself to be well suited to the development of autonomous behaviour within a virtual world simulation. The model provides a kind of encapsulation of parallelism, because the system's state is truly distributed. "…a model of computation based on the notion of actors, active objects that communicate by message passing." and further, "Actors blur the conventional distinction between data and procedures." [Lie87, p. 1]

5. Conclusion

The SAME4KIDS project is in its early stages and certainly a lot have to be done. However, it is encouraging to know that the project has been received enthusiastically by two different kindergartens, which showed there is a need for such tools. The data gathering from the prototype experiments will show if children can learn simple words in other languages. For example, the project will register if a kid made the right or wrong choice for different language commands.

With respect to the implementation, we intend to use the application of language learning as a series of prototypes for further developing the *Animarium* environment as a virtual laboratory, with uses in both research and learning situations.

In addition, we wish to test the SAME4KIDS language learning games into the realm of immersive projection and speech recognition. *Animarium* could be adapted to support both technologies with relative ease, and the effects of a more intuitive and immersive interface on the learning level of the target audience subsequently studied.

References:

[Hew76] Hewitt, Carl: Viewing Control Structures as Patterns of Passing Messages (1987). `http://www.cypherpunks.to/erights/history/actors/AIM-410.pdf`

[Lie87] Lieberman, Henry: Concurrent Object-Oriented Programming in Act 1. *Object-Oriented Concurrent Programming, A. Yonezawa and M. Tokorro, eds., MIT Press, 1987*

[Pap93] Papert, Seymour: The children's machine. Rethinking School in the Age of the Computer. *Basic Books , 1993*

[Mil99] Milner, Robin: Communicating and Mobile Systems – The Pi Calculus. *University of Cambridge, 1999*

[Tok01] Tokuhama-Espinosa, Tracey: Raising Multilingual Children. *Greenwood Publishing. 2001*