Advanced Authoring Tools for Game-Based Training

A. Tanju Erdem          Bora Utku          Tolga Abacı          Çiğdem Ergöz Erden*†

Momentum Digital Media Technologies, TUBITAK MAM TEKSEB, Gebze 41470 Kocaeli, TURKEY
*Bahcesehir University, Electrical and Electronics Engineering Department, Beşiktaş 34353 Istanbul, TURKEY
{terdem, butku, tabaci}@momentum-dmt.com, cigdem.erdem@bahcesehir.edu.tr

Keywords
Game-based training, non-linear story telling, real-time GPU rendering, 3D content authoring, speech-driven face animation.

Abstract
We present a complete system of authoring technologies for creating scenario-driven 3D games for training. The proposed system includes advanced tools for authoring and debugging non-linear training scenarios, creating and real-time rendering of immersive 3D environments, and automatic generation of character facial animations using recorded speech. The scenario authoring tools make it possible to easily construct and debug complex training logic flows. The 3D real-time rendering tools enable content developers with a what-you-see-is-what-you-get working environment and hence greatly reduce the production time for 3D scene and character development. The speech-driven face animation tools provide for automatic generation of lip and emotion animation.

1. INTRODUCTION

Lack of properly trained troops and commanders can have devastating consequences in today’s world of technologically complex global conflict. Game-based training solutions have been recognized as effective and cost-efficient alternatives to conventional live training, especially in military training environments [1,2]. The game-based approach can be applied as “virtual simulation training” provided by networked immersive 3D simulators, or “constructive simulation training” using scenario-based large-scale battle models [3].

We present a complete system of technologies and tools for building scenario-driven game-based training simulations. A key component of this system is an integrated development environment (IDE) for authoring and debugging non-linear scenarios for training games.

The system also includes a visualization module for real-time rendering of immersive 3D environments, and automatic generation of character facial animations using recorded speech. Either the whole training system or a subset of its modules can be applied to a wide range of game-based simulation solutions. Figure 1 is an overview of the game-based training system.

We discuss the architecture of the IDE in the next section. The real-time rendering aspects of our work are discussed in Section 3. We then explain our innovative methods for speech-driven facial animation in Section 4. We conclude by a summary of potential benefits and uses of this new training simulation development system in Section 5.

Figure 1 Authoring system overview

†Formerly with Momentum Digital Media Technologies
2. NON-LINEAR SCENARIO BUILDING

We have developed an IDE that is specifically aimed at scenario-driven game-based training. A typical IDE is used for software development and normally consists of a source code editor, a compiler and/or interpreter, build automation tools, and usually a debugger [4]. An IDE can be used for designing computer processes as a non-linear workflow assembly [5]. An IDE can also provide an environment for writing and modifying computer program source code directly with graphical means [6]. Our IDE is also graph-based and designed to be used by non-programmer users who are involved in scenario generation processes. It has a tightly integrated combination of corresponding modules: A visual programming language (VPL) and its respective logic engine (interpreter), a scenario authoring tool (source editor), and a scenario testing and debugging tool (debugger).

2.1. Logic Engine based on Visual Programming

A VPL lets users specify programs by manipulating program elements graphically rather than textually. It allows programming at a relatively higher conceptual level, with visual expressions, spatial arrangements of text and graphic symbols like boxes and arrows drawn in specific colors, styles, etc. The VPL concept is similar to dataflow programming paradigm, as opposed to conventional imperative programming [7]. A recent visual dataflow programming language is Microsoft Visual Programming Language (MVPL) in Microsoft Robotics Studio, aimed for robot control and simulation programming [8].

We have developed a VPL suitable for constructing non-linear stories. In our VPL, a language interpreter tool has been implemented as the logic engine. Our VPL encodes a non-linear logic flow in the form of a logic flow graph. The nodes of the graph represent logic statements and the links of the graph represent types of actions and flow of the logic state data. There is a set of node types and link types, each corresponding to a specific action like creating parent-child hierarchies, setting logic flow directions, setting priorities, enabling/disabling other nodes, iterating over linked nodes, switching based on logic conditions, user choices or random choices, and running command scripts, etc. All those actions relate to high-level non-linear story flow design elements, thus anyone who is capable of designing a story or script using prose will find it natural to recreate that design in this graphical and machine executable representation. Unlike common programming
language interpreters, our interpreter directly executes the graphical program definition stored in a binary format.

**2.2. Scenario Authoring Tool**
The Story Authoring Tool is a Graphical User Interface (GUI) module that has been developed with the main goal of allowing non-programmer users to create non-linear story flows with the VPL. As the “graphical source editor” of the VPL, this tool allows creation, modification and viewing of VPL programs through mostly editing of graphical language components in a drawing canvas, instead of typing code into text files with the keyboard as can be seen in Figure 2. Main functionalities include typical authoring actions like browsing, searching, saving, retrieving, and verifying VPL programs. All intermediate data and output data are stored in a relational database. Any partially or fully completed scenario can be immediately transferred to the Testing and Debugging Tool for design verification and logical consistency testing.

**2.3. Testing and Debugging Tool**
This is a GUI module that allows debugging the VPL programs created by the Scenario Authoring Tool. Common programming languages allow for many types of errors to go unnoticed during development phase, so they require a debugger tool to detect low-level programming errors. Debuggers use methods like execution by stepping through the code or execution by special configurations. Since our VPL structure and authoring tool does not even allow for low-level structural mistakes, the debugging tool focuses on catching logical errors. This tool uses the language interpreter to execute a VPL program with functions to step through the high level logical flow of the program and view the current values of high level graphical data structures. As an example, a VPL program allows nodes to be dynamically enabled or disabled during execution. A scenario is possible in which a node in the story graph will not lead to any other valid nodes because they have been disabled in previous execution steps. Hence, logically the program will come to a dead-end if the user ever gets to that point in the story. The debugger helps users catch such problematic cases. Testing can be limited to a subset of a given program graph by marking certain nodes as critical. Another novel feature of this testing tool is the ability to randomly play and test a given scenario. Running a random test is an efficient way of discovering possible dead-ends in a scenario.

3. **REAL-TIME RENDERING**

Our GPU-based real-time rendering engine is capable of rendering immersive, realistic 3D virtual environments efficiently. The engine has been integrated as plug-ins into Autodesk Maya™ to facilitate authoring of content, and into the OGRE to enable standalone real-time rendering.

3.1. **The Rendering Engine**
The rendering engine is based on DirectX 10. The choice of the rendering API is primarily motivated by clean and tidy support for the new features introduced by the latest generation hardware. Better driver and debugging support compared to OpenGL have also played a role.

Contrary to most contemporary GPU-based real-time rendering engines, we have chosen not to sacrifice rendering quality for performance. However, this does not mean that high-performance real-time operation has been overlooked. In addition to the usual Blinn-Phong shading model, we have incorporated several material models into the engine, including parallax occlusion mapping [4], hair and skin shading with advanced subsurface scattering [10]. The engine allows for multiple point/spot/directional lights to affect each shaded object. The engine performs real-time shadow rendering [12,13] that is of better quality than those generally seen in contemporary game engines. Our shadow mapping technique uses a resolution enhancement algorithm that significantly increases the effective resolution of shadow maps and this results in a marked reduction in "jaggies" caused by shadow map aliasing. In addition, rendered shadows are filtered in screen-space so that any remaining aliasing problems are visually compensated for. Real-time light shaft rendering with shadows [14] is also supported. In addition, basic indirect illumination can be achieved by using cube/dual paraboloid maps [12] that approximate glossy environment reflections.

The engine implements High Dynamic Range (HDR) rendering with anti-aliasing [16] and several post-processing effects including bloom and depth-of-field [17]. Atmospheric effects such as fog and rain [18] can also be utilized. Some other state-of-the-art features of interest provided by the engine are: Real-time reflection effects (mirrors), GPU-accelerated skinning and blend-shape animation, and transparency handling with alpha-blending and sorting. Sample screenshots of scenes rendered with our engine can be seen in Figure 3.

3.2. **Content Authoring with Bee™**
In order to generate content for a 3D virtual environment, industry standard tools such as Autodesk Maya™ [19] or 3D Studio MAX [20] can be used. These tools provide advanced capabilities that enable artists to create models and animation. However, little provision is usually made for the real-time shading aspect of content authoring. Artists usually do not have the sufficient means to preview how the shaded models will appear in the virtual environment. Although modeling and animation packages provide some hardware shading facilities to display how a model will look when a given shader is applied to it, these are seldom enough. The final appearance of a shaded object depends not only on the shader but on the many post-
effects applied or other full-scene effects such as projective textures or shadows. Some existing commercial game engines [21,22] try to overcome this problem by introducing their own “editor” tools in the content generation pipeline. With this solution, models must be created first and then imported into the “editor” for previewing the shaded model and setting up of shading parameters. We argue that this decoupling impairs efficiency and disrupts the workflow for the artist.

We propose to integrate the real-time rendering engine into the modeling tools to solve the problems outlined above. As a proof of this concept, we have created a plug-in for Autodesk Maya™ called Bee™ that fully integrates our rendering engine as a custom viewport renderer. Bee™ allows the graphics artists to instantly see the effects of changes in shader parameters and textures, and to observe effects of advanced high dynamic range post-processing effects and real-time lights and their shadows. Existing scenes can be converted for Bee™ real-time rendering with ease.

Figure 4 shows Bee™ in operation.

3.3. Standalone Real-time Rendering with OGRE

The end user of our game-based training system interacts with a standalone real-time application. For maximum flexibility, this application needs to have a modular architecture that allows custom subsystems to be plugged-in dynamically. In addition, the application must be based on cross-platform solutions for maximum availability. In order to meet these goals, we have selected the OGRE open source graphics engine [23] as the base of the visualization module. The OGRE engine and its plug-ins provide an extensible, cross-platform framework that allows us to develop standalone applications for various platforms, without re-implementing common tasks like input handling, user interfaces, scene management, audio management, physics, logging, etc. The rendering system of OGRE engine has been customized to provide the same features with our own Bee rendering engine, which allows a perfect visual match between the modeling tools and the final application. Our logic engine integrates seamlessly with the standalone application. The application is also linked with a TCP/IP based networking module and a navigation/path-finding subsystem.

4. SPEECH-DRIVEN FACE ANIMATION

In training simulations, realistic 3D avatars are needed to conduct dialogs for efficient and realistic human-computer interaction. Recently, technologies for generating a 3D head model of a person using his/her photographs have been developed [24] and successfully applied to applications such as PC adventure games and mobile communication [25]. Natural looking lip and emotion animation, synchronized with incoming speech, is also essential for realistic character animation.
Audio-visual mapping models for facial animation have been presented in the literature [33-35]. These models require video analysis prior to facial expression synthesis in order to find a mapping from audio patterns to facial motion trajectories. In this work, our goal is to build a speech-based viseme and emotion recognition system to drive fully automatic facial expression synthesis. In Section 4.1, we evaluate several approaches for generating realistic “lip synchronization” using Hidden Markov Model (HMM) based recognition systems. In Section 4.2, we present a speaker-independent framework for automatically generating the “facial expression animation” of 3D talking heads using only the speech information.

4.1. Speech-Driven Lip Animation

Humans are very sensitive to the slightest glitch in the animation of the human face. Therefore, it is necessary to achieve realistic lip animation which is synchronous with a given speech utterance. In this section, we present a language and speaker-independent approach for lip synchronization. We compare four different acoustic units within HMM structures for generating the viseme sequence to be used for synchronized lip animation. These acoustic units are namely phone, tri-phone, viseme and tri-viseme based units. Our lip animation method is based on 16 distinct viseme classes [31]. After the generation of the 3D head model [24], a graphic artist defines the mouth shapes for the 16 visemes using a graphical user interface. Sample visemes corresponding to various phoneme classes are shown in Figure 5. Since speech has both an auditory and a visual component, it is very important that the definitions of visemes are done accurately. These mouth shapes (visemes) are properly interpolated (smoothed) during the actual animation.

A phone is the basic acoustic speech unit. We use the TIMIT speech database [32], which has 46 defined phonemes for American English. In an attempt to increase the recognition rate, we also tested tri-phone based HMM structures to generate context-dependent models. A tri-phone is a phoneme with left and right context (left phoneme – phoneme + right phoneme).

For lip animation, it is not crucial to recognize the exact phonemes composing the speech. It is sufficient to recognize the visual components, i.e., visemes. We aim to
obtain acceptable viseme sequences comparable to phone-based method by narrowing the set of acoustic units to 16 viseme classes. We also evaluated the tri-viseme based acoustic units. Analogous to a tri-phone, a tri-viseme is also a context-dependent structure: a viseme with left and right context makes a tri-viseme.

We use the TIMIT corpus to build speaker-independent HMM models and to generate appropriate language models corresponding to each model unit (phone, tri-phone, viseme and tri-viseme based units). We observed that the recognition rate of the tri-viseme based HMM method is the highest (79.49 %), which is followed closely by the tri-phone based HMM method (78.75 %). Viseme and phone based acoustic units yielded recognition rates of 73.01 % and 68.36 % respectively.

Figure 5 Example visemes for phoneme classes. Viseme 2 corresponds to the phoneset (ay, ah), viseme 3 corresponds to the phoneset (ey, eh, ae) and viseme 4 corresponds to the phoneset (er).

4.2. Speech-Driven Emotion Animation

Facial expressions and the accompanying speech are correlated with emotional states of a person. Thus, we aim to build a correspondence model between emotional speech parameters and facial expressions.

Although extensively investigated [26,27,29], recognizing emotions for computers is still an open problem. The aim of this sub-section is to investigate different spectral and prosodic features, mixture of different features and fusion of different classifiers for better emotion recognition and better speech-driven facial expression synthesis. In this investigation, we use GMM based emotion classifiers to model the color of spectral and prosody features, and HMM based emotion classifiers to model temporal emotional prosody patterns.

Speech is the only input modality that drives our automatic emotion recognition and facial expression synthesis system. The overall system is trained and tested on the Berlin Emotional dataset (EMO-DB) [28] which contains speech utterances reflecting the seven emotions, namely, happiness, anger, fear, sadness, boredom, disgust, and neutral.

4.2.1. Feature Extraction

It is well known that for different emotional states, the speech signal carries different prosodic patterns [29]. Hence, prosodic features such as pitch and speech intensity can be used to model different emotions. For example, high values of pitch appear to be correlated with happiness, anger, and fear, whereas sadness and boredom seem to be associated with low pitch values [29].

In the training part of our system, first the emotional speech data is parameterized using the short-term acoustic features. These features include spectral features, such as mel-frequency cepstral coefficients (MFCC), line spectral frequency (LSF) features and their dynamic features (i.e., the first and second derivatives), as well as prosody-related features consisting of mean normalized pitch, the first derivative of pitch, and intensity.

Although some of these features have been recently employed for emotion recognition, our investigation includes the following novelties: (i) we use LSF features, which are good candidates to model prosodic information since they are closely related to formant frequencies, (ii) we employ a novel two branch HMM structure to model two classes of temporal prosody patterns, which are expected to be related and unrelated to the underlying emotion, and (iii) we investigate data fusion of different features and decision fusion of different classifiers, which are not well studied for emotion recognition framework.

4.2.2. Classification and Decision Fusion

We use GMM-based emotion classifiers to model the probability density distribution of spectral speech features, and HMM-based emotion classifiers to model the temporal emotional prosody patterns.

In the GMM based classifier, probability density function of the feature space is modeled with a diagonal covariance GMM for each emotion. Probability density function, which is defined by a GMM, is a weighted combination of $K$ component densities.

In the HMM based classification, we model the temporal patterns of the emotion-dependent prosody contours through HMM structures. The HMM structure is set to two parallel branches, where each branch has $N$ left-to-right states with a possible loop back. One can expect that one of the branches models emotion dependent prosody patterns in the utterance and the other branch models emotion independent patterns. We use pitch, the first derivative of pitch, and speech intensity for HMM classifiers to model the prosody patterns.

We consider a weighted summation based decision fusion technique to combine different classifiers [30]. The GMM and HMM based classifiers output likelihood scores for each emotion and utterance. After normalization [30], we have two likelihood score sets for GMM and HMM based classifiers for each emotion and utterance. The
decision fusion then reduces to computing a single set of joint log-likelihood ratios for each emotion class. Assuming the two classifiers are statistically independent, we fuse the two classifiers, by computing the weighted average of the normalized likelihood scores.

4.2.3. Experimental Results

In the experimental studies over the Berlin Emotional Speech dataset, the proposed method achieves an average emotion recognition rate of 83.42% using a GMM classifier of MFCC and dynamic MFCC features. Moreover, decision fusion of two GMM classifiers with MFCC and LSF features yields an average recognition rate of 85.30%. Also, a second-stage decision fusion of this result with prosody HMM further advances average recognition rate up to 86.45%.

In the facial expression synthesis part of the proposed system, first, speech-driven emotion recognition is carried out using the previously trained GMM and HMM classifiers. Then, the corresponding facial expression among the seven emotions that are predefined by a graphic artist on our 3D head model is animated. As the recognized emotion changes during speech, the changing facial expressions are linearly interpolated in time. Several samples of synthesized emotional facial expressions are given in Figure 6.

![Figure 6 Synthesized facial expressions for fear, happiness, anger, and sadness (from left to right)](image)

5. CONCLUSION

The training system presented in this paper combines our novel contributions from various research areas as logic programming with visual languages, advanced graphics rendering pipeline design, and speech-driven face animation. Our visual programming based logic engine design approach greatly simplifies the task of building non-linear scenarios for game-based training systems. Our real-time rendering engine utilizes the programmable architecture of modern GPUs to provide high-quality rendering results to both 3D content artists and end user trainees. Finally our facial modeling and animation technology allows us to create photo-realistic 3D head models of real people and animate them realistically using speech.

References

[6] USPAT6792595 - Storistenau, Adrian (Toronto, CA), Weiss, Robert (Brampton, CA) 2004 Source editing in a graphical hierarchical environment United States IBM (US). U.S. Patent No. 6,792,595

**Biographies**

A. Tanju Erdem received his Ph.D. in electrical engineering from the University of Rochester, Rochester, USA, in 1990. He is currently the CTO of Momentum Inc., a technology SME developing core technologies and products for 3D animation and game-based entertainment and education. Prior to Momentum, Dr. Erdem was with the Research Laboratories of Eastman Kodak Company, USA (1990-1998). He was a member of the MPEG Committee (1992-1998) where he has served as the chairman of the MPEG-2 ad-hoc group on 10-bit video compression (1993). He is a member of IEEE (1988-) where he has served as the President of the Rochester Chapter of the IEEE Signal Processing Society (1992-1994). He holds 7 U.S. patents and has authored or coauthored more than 50 technical publications in the field of video processing and computer graphics.

A. Bora Utku received his M.Sc. in computer engineering from the Boğaziçi University, Istanbul, Turkey, in 2001, and his MBA from Yeditepe University, Istanbul, Turkey, in 2008. He has been an active R&D software engineer for the last decade, working on visual simulation, military training, and entertainment projects. He is currently a project leader in Momentum Inc., a technology SME developing core technologies and products for 3D animation and game based entertainment and education.

Tolga Abacı received his Ph.D. in computer graphics and virtual reality from the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 2006. He has been an active researcher and has authored or coauthored several technical publications in the fields of real-time rendering and animation, face modeling and virtual reality. He has participated in several EU research projects. He is currently a project leader in Momentum Inc., a technology SME developing core technologies and products for 3D animation and game based entertainment and education.

Çiğdem Eroğlu Erdem received her Ph.D. degree in Electrical and Electronics Engineering from Boğaziçi University, Turkey, in 2002. From September 2000 to June 2001, she was a visiting researcher in the Department of Electrical and Computer Engineering, University of Rochester, NY, USA. During 2003-2004, she was a postdoctoral fellow at the Faculty of Electrical Engineering at Delft University of Technology, the Netherlands, where she was also affiliated with the video processing group at Philips Research Laboratories, Eindhoven. She has published more than 30 technical papers in the areas of digital image, video and speech processing.