Assisted Animated Production Creation and Programme Generation

Juan Abadia, Alun Evans, Eduard Gonzales, Sergi Gonzales, Daniel Soto, Santi Fort, Marco Romeo, Josep Blat
Universitat Pompeu Fabra – Fundació Barcelona Media
Tanger 122 - 140
08018 Barcelona, Spain
+34 93 542 20 00

alun.evans@upf.edu josep.blat@upf.edu

ABSTRACT

The creation of animated productions is a labour intensive process. Whether the end result is a large-budget motion picture, or a small-scale internet production, there is invariably a large amount of time spent in creating the timeline, arranging assets, previewing and editing. This iterative process is necessary in large-scale productions but can become repetitive and frustrating when the end result is a small production that may have similar elements to previous work. We present a workflow system and framework that are able to both greatly facilitate animated programme production and introduce an element of procedural generation. We further present the Programme Editor, an application designed to be a powerful front end for the framework. The principal contribution of this work is the creation of an XML-based scripting engine that can be used to create an animated production. This permits several techniques, tools and workflows to interchange information, allows rapid incorporation of further tools, and furthermore facilitates the complete automatisation of the production process.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Resuable software – *reusable libraries, resuse models.*
H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems – *animations.*
H.5.2 [**Information Interfaces and Presentation**]: User interfaces – *Graphical User Interface, Prototyping.*
I.3.8 [**Computer Graphics**]: Applications.
J.7 [**Computer Applications**]: Computers in other systems – *Publishing, real time.*

General Terms

Performance, Design, Human Factors.

Keywords

Reusable content, multimedia automation.

1. INTRODUCTION

Since the idea of convergence arose in the early 1990s, the media industry has looked at cross-platform exploitation methods as a way of producing more exciting content, more cost-effectively. Yet while technology has helped to produce better quality sounds and images, the costs continue to rise. Digital media production remains very labour intensive, making it a very high-risk, high-cost industry. One of the reasons is that productions are crafted, almost without exception, at very low levels, in order to better satisfy artistic needs. Indeed, in many applications, the existence of more sophisticated digital tools has actually pushed up costs, as more time is spent on complex off-line processes in the quest for quality.

This leads to the concept of "re-usable" media, content that can repurposed and used again for a different production. A classic example of this has existed for years, with films and television productions being dubbed into different languages, but what about situations where the actual *content* of the production should change? This is the situation that is addressed in the paper, with the presentation of a framework for assisted creation of animated productions and the automatic generation of programmes that vary in visual and audio content.

The work presented can be split broadly in two aspects: the first is introduction of the multimedia production scripting *framework* which allows fine control of the creation of an animated production or clip. The principal contribution of this work is enabling aspects of animated production to be *automated* and *re-used* according to external data. The framework also allows multiple reuse of assets and semi-automatic programme creation, and the complete separation of content from rendering.

The second aspect deals with the creation of an advanced editing application, the 'Programme Editor' for rapid prototyping and programme setup, and designed specifically for use with the framework. The Programme Editor is capable of rapidly adjusting different aspects/sections of a programme (for example, camera type, actor locations, audio, and provide on-the-fly previewing of the results. While essentially acting as a front end for the programme scripting and generation aspects of the framework, the Programme Editor is nevertheless a powerful application in its own right, using a custom 3D graphics engine to display results in real time (see Figure 1).

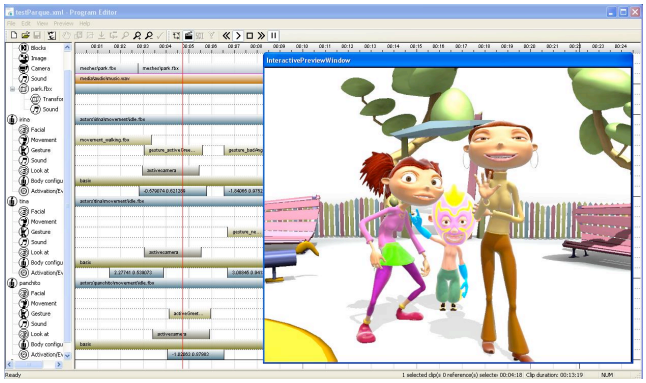

Figure 1: The Programme Editor and real-time preview window

The paper is structured in the following order. Following a discussion of related work we present in detail the framework for assisted programme generation. We then present the Programme Editor application, before providing several example of the use of our work in both the commercial and academic fields. Finally we conclude that the presented work is a novel and powerful methodology for assisted animated production creation and programme generation. The professional use that the framework has already seen suggests that it provides real benefit to production workflow, and provides a strong platform for the integration of novel multimedia technology

2. RELATED WORK

Perhaps the most similar work to our framework is the Synchronized Multimedia Integration Language (SMIL) [1]. SMIL is a W3C recommended markup language for describing multimedia presentations. It is written directly in XML, and defines markup for layout, animation, timing, visual transitions and embedded media such as text, audio, images and video. SMIL is the basis for the more famous Multimedia Messaging Service (MMS), which is now ubiquitous within mobile telephony, and was one of the underlying technologies for the now defunct HD-DVD format. SMIL, and the possibilities of its use, were a strong inspiration for the work in this paper, which can be thought of extending SMIL to provide more complete cover for animated production.

In terms of video editing applications, there are several companies that provide similar instant editing capabilities. Redboard [2] is storyboarding and prototyping technology, designed to enhance existing workflows by integrating the traditional skills of the storyboard artist into a CGI workflow. Their system consists of a powerful computerised storyboarding tool coupled to a CGI renderer, allowing artists to transfer their ideas directly to the screen. Reboard estimate that the increased communication between directors, producers and artists permitted by their system leads to a reduction of up to 40% in storyboarding costs. Redboard has already underpinned in-house productions, and it is now launching for licensed use.

A slightly different approach is taken by French company Xtranormal [3]. Instead of targeting high end users, they have developed an online system that allows novice users to "make movies", using several preset combinations of backgrounds and characters. The browser-based technology allows users to create simple 3D animated clips with multiple characters, employing text to speech software to provide audio files for the characters to utter. Once configured, the clip is rendered and downloaded to the browser for viewing and sharing on online video sites such as YouTube.

Despite the technological accomplishments of commercial software, there is a lack of automation and possibilities for connection to external data. Indeed, the task of automatic animated production generation is not that has seen that much previous work in the academic field either. Assisted video editing, of course, is a field that has seen much work, with Girgensohn *et al.* [4] providing a recent example of advances in the field, backed up by powerful commercial applications such as iMovie [5] and Premiere [6]. For a more theoretical overview of usability of visual programmeming environments, the reader is directed to Green and Petre [7].

3. WORKFLOW OVERVIEW

3.1 Programme Scripting via XML

The XML scripting framework, while designed from scratch, was inspired by the scripting philosophy behind SMIL [1], particularly with regards to self-reference of contents with respect to a time-line. The main difference is that the scripting XML explained in this section makes use of the concept of different "content tracks" that allow relative referencing (i.e. not direct referencing in time). This allows the system to link any content without restriction. In SMIL these concepts do not exist and are approximated with production flows which lack the same flexibility.

The introduction of this relative referencing means that our scripting framework is much more suitable for programme scripting than SMIL. In this section, we describe in detail the different elements that are used within the framework.

3.1.1 The Timeline

The workflow scheme we have defined for the generation of programmes describes such programmes as a set of clips and relations between clips. Each of these clips and relations are defined using XML, and placed with reference to a common base *timeline*. The timeline is the basis of the programme and used to tie the programme together into a coherent whole.

3.1.2 Programme Components

Each clip represents a *component* of the programme, examples of which are:

- camera location and direction
- virtual character location and orientation
- animation clip (for any component, including cameras)
- audio file
- background image

Each component can therefore be regarded as its own individual entity; it can be placed at any location along the timeline and can be moved forwards or backwards in time, simply by changing one variable within the XML template. Furthermore, due to the fact that each clip is merely an XML file, once generated it can be shared and used among different programmes. Thus it is possible to build up a variety of generic clips that can be re-used for several purposes.

3.1.3 Time Dependencies and Relations

The relations between the components are also defined using XML (with reference to the base timeline). The relations create a series of *time dependencies* that indicate time constraints between the components. This ensures that there are no clashes between similar types of component, while allowing components that do not affect each other exist at the same point along the timeline. For example, one clip may specify that, between the time t=10s and t=20s, the virtual camera should look at a certain point in the scene. This *relation* for this component specifies that, during the same period of time, it is perfectly possible for an audio component to exist at the same point on the timeline; however, another camera may not overlap.

3.1.4 Independence

The main advantage of the scripting schema that we have developed is that it is completely independent of the rendering of the programme. This allows several important benefits when programme rendering is considered

- A programme description can be rendered to different resolutions and qualities, yet still remain the same programme. The programme description specifies *what* is happening, not *how*.
- The time dependencies indicate time constraints between actions.
- It is possible to change the representation of the components and obtain the same programme with different assets. For example, a programme description might indicate that there is a component of type *actor,* which at time t=10s raises a hand. The current programme might also specifies the actor is called 'Actor1', and thus use meshes, textures and animation from 'Actor1'. Yet if we wish to swap to a different actor, there is no need to re-script the programme. It is possible to change the name of the actor to 'Actor2' and still keep the fact that at time t=10s s/he will raise her/his hand. The visual representation of the actor and how each actor raises the hand is kept independent of the programme design stage, and moved to the rendering process.

3.1.5 Block Templates

As described above, the work of facilitating the process of programme generation is based around the concept of defining a set of components and the time dependencies between them. *Block templates* assist further with this facilitation by grouping components together and allowing for modification according to some external input. For example, a block template for a programme showing today's weather might consist of an actor, background and camera, linked on the timeline along with the weatherman character's associated gestures and audio files to explain the weather.

The principal advantage of saving configurations of components in this manner is that the individual components can be changed according to *external* factors, thus creating a different programme based on data that may come from database, or from user input. In the example of the weatherman, typical components that could be changed according to the current weather forecast would be the background image of the scene and the audio file that the actor would recite. A further example is that of a user interaction – users could change the appearance of a character, or write a sentence for the character to utter (via a text-to-speech engine). Further possibilities include the ability to dynamically change actor gestures or facial expressions depending on the input (although this would require artificial intelligence beyond the what the framework currently offers).

The framework also supports the use of changing variables within block templates, for example for positioning objects within the scene. These variables can be set to be constant, or vary along certain paths, or even to be completely random. An example may be a bird flying through a scene, and taking a different route each time the scene is replayed.

In summary the grouping of components into block templates allows the output of the framework to be *dynamic*. In this sense it is a dynamic scripting framework, allowing the possibility for programmes to be scripted while retaining (and encouraging) the change in variables and components according to external factors. The fact that the framework is based around a well-defined XML schema allows such changes in scripting to be carried very easily, as is described in Section 3.2, and demonstrated in several of the use cases described in Section 5.

3.2 Programme Creation and Editing

There are four main methods of creating a programme or scene using the framework. The most basic method is manual creation. In this sense the framework can be thought of as an extremely high level scripting language, capable of creating scenes, animations and interactions along a time line. While manual creation offers precise control over a scene, the disadvantage is of course the time involved in writing code by hand.

A more powerful option is the use of software designed specifically to replace the need for manual coding. The requirements for this software are that it should be intuitive enough to be used without the need for detailed training, yet be powerful enough to be able to modify scenes to a high level of detail. To this extent we have developed the Programme Editor, a powerful yet straightforward visual timeline editor. The Programme Editor is explained in detail in Section 4.

An extension to the Programme Editor is the possibility of creating a web-portal for modification of a scene. For example, once a block template is created using the Programme Editor, the variables within it can be changed by users, via a web-portal, and the results encoded as a video and downloaded by the user. An example of this has been mentioned above, that of using a text-to-speech engine to allow users to specify an utterance for a character (see Section 5.2.2).

Finally, the framework allows easy integration with more powerful visual editors such as 3D Studio Max (see Figure 2), allowing the exporting and use of assets and textures in a straightforward manner.

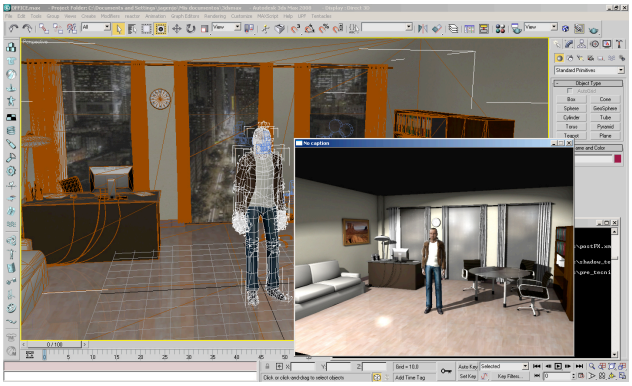

Figure 2: 3D Studio Max direct export to the framework

3.3 Visualisation

Once the programme description is generated, it is then passed to the rendering software for visualisation. Visualisation takes place either in real-time or as a video for various devices.

3.3.1 Real Time Visualisation

The framework comprises a powerful shader-based rendering engine with sophisticated culling and lighting effects. The engine supports real-time update of a scene, for example if a component is added to a scene, or an existing component modified, the scene description is seamlessly reloaded and the component appears within the scene. This is particularly useful when prototyping how a scene will appear, as it enables rapid addition/modification of components without the need for the engine to reload each time. For example, character appearance or gestures can be modified quickly in order to quickly deduce which are more appropriate for the scene; or random changes of variables can be quickly tested to ensure that they are coherent. Furthermore, this visualisation allows real-time interaction with the scene, in the form of users being able to move objects and the camera (although this is not so relevant to the generation of programmes).

The real-time renderer works via OpenGL. It is available both as a direct executable and as a plug-in for internet browsers (such Firefox and Internet Explorer), and runs on a variety of platforms (included handheld platforms such as the iPhone).

3.3.2 Video Generation

Using the graphics engine, the framework can generate videos based on the programme description. As mentioned above, this area houses one of the most powerful capabilities of the programme generation framework, as it is completely separate from any decisions regarding content. This allows videos to be generated for a variety of different platforms and delivery methods. Videos can be encoded using different codecs, using different compression algorithms, and at different resolutions. This capability is of high importance given the variety of the use cases presented in Section 5 (from television broadcast quality, to downloadable flash video).

4. THE PROGRAMME EDITOR

This section intends to elaborate on the 'front end' of the framework – the application that has been developed specifically to drive assisted programme production using the framework. The *Programme Editor* is a Windows application that provides an intuitive drag-and-drop style interface to the placement of programme components in the relation to both themselves and the timeline.

4.1 Basic track/timeline structure

The structure of the Programme Editor is representative of the structure required by the framework to create a programme description. A programme consists of a series of base *tracks* along the timeline, to which components can be added. Each track is responsible for containing components of a particular type: the background image, background objects and terrain, camera position and animation, and any ambient audio. To the basic set of programme tracks, *actors* can be added. Each actor has associated with it its own set of tracks that are independent from the main programmes tracks. The list of tracks is displayed on the left hand side of the software, with the timeline running from left to right, and displayed above. Figure 3 shows a screenshot of the structure of a programme without any components added. The left pane shows the list of tracks, while the right pane shows the timeline and contents of the tracks.

4.2 Component addition and movement

Addition of components to a track is simply a case of right-clicking anywhere on the track and using a drop down menu to place components. The menu will only show assets and components that are compatible with that track, regardless of the current data directory (see Section 4.5). Once the component has been selected, it will appear as a coloured rectangular *indicator* on the track, overlaid with the name of its source component. The location and lateral length of the indicator on the track, with reference to the timeline, defines the start and end point of its effect within the programme. For example, if the indicator for component "Camera1" lies on the track between 00.10" and 00.15" on the timeline, between these times the camera used by the programme will be as specified by component 'Camera1'. By holding the left mouse button on the indicator, it can be dragged and dropped to any point along the timeline, although it cannot overlap with another camera. By click-dragging either end of the indicator left or right it is possible to change the duration of the effect, and by right-clicking on the indicator it is possible to rapidly align it to the boundary of the previous and/or following clip (or the start of the scene). Furthermore it is possible to alter the 'fade in/out' property of certain components so that their effect does not appear/disappear immediately (this is more relevant to animation and audio components).

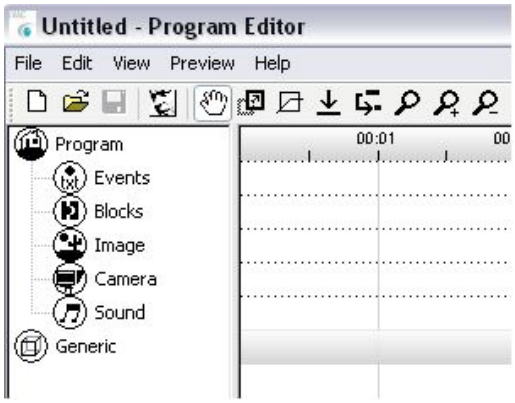

Figure 3: Basic programme structure

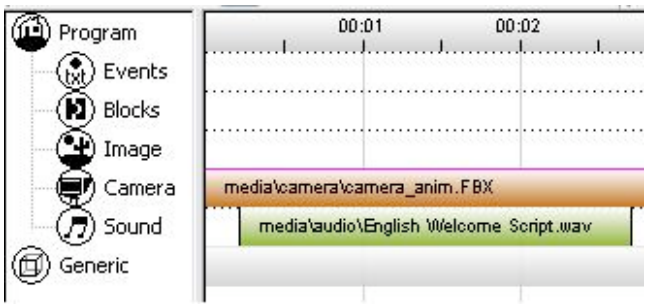

Figure 4: Camera and audio components added to programme description

Figure 4 shows an example of a camera component added to the timeline, and an audio file added below it. During this region of the programme, the output will be viewed using this particular camera, and the background audio will be played from this particular audio file. Furthermore a background scene, complete with light sources, can be loaded into the left pane to provide context for the programme.

4.3 Actors

Virtual characters located within a scene are a prominent feature of the framework and Programme Editor. Within the left pane of the editor, a sub-menu allows the addition of an actor to the scene. The number of actors added is limited only by the rendering capabilities of the machine that is running the software (which itself is determined by the visual complexity, for example the number of polygons, that each character has). Once a character is added, the left pane displays the tracks for that character, as shown in Figure 5. The basic tracks of an actor consist of:

- *Default* – specifies the default whole-body animation of actor, usually an idle pose or animation.
- *Facial animation* – indicates the current facial animation of the character
- *Movement* – specifies the current whole-body movement animation of the character (i.e. animations that will change the location of the character in the scene).
- *Gesture* – defines any gesture animation (i.e. an animation that will not alter the location of the character) e.g. a waving hand.
- *Sound* – indicates the current speech file that is uttered by the character. The software uses simple babble loop animation to produce basic but effective lip-synch.
- *Look-at* – specifies where the character should be looking. Can be the active camera, another object within the scene, or another character.
- *Body configuration* – specifies different configuration of changeable items on the character e.g. clothes.

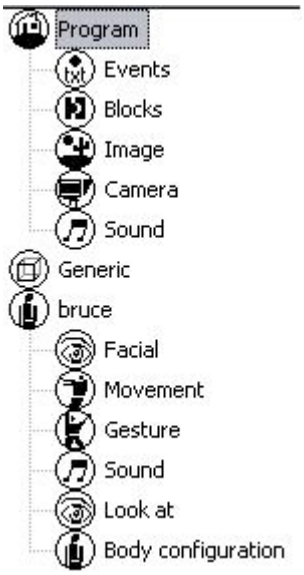

Figure 5: Addition of character "Bruce" to the scene

Figure 6 shows a typical screenshot of two characters with several components assigned to their tracks. Both characters are looking at the current active camera; one ("irina") is currently walking forward, the other ("panchito") is making a greeting gesture while uttering a text file. The drag and drop nature of the components means that it is straightforward to adjust the location and/or duration of any of these components, and once the programme setup is saved in XML, other software could modify the code to change the data source for each component (for example, to change the source for Panchito's utterance to a different audio file).

There are several important parameters for the actors' appearance and behaviour that can be changed via context menus. An actor's position and orientation can be changed, and the loop/stretch status of individual animations can be changed. For example, if a character is to act a 'wave' animation for five seconds, yet the actual duration of the animation is only three seconds, the user can choose whether the animation is looped or stretched (slowed) to accommodate the extra time. Importantly, the fade in/out tool (mentioned above) can be used on all animations to ensure that actors do not suddenly jerk unnaturally into movement.

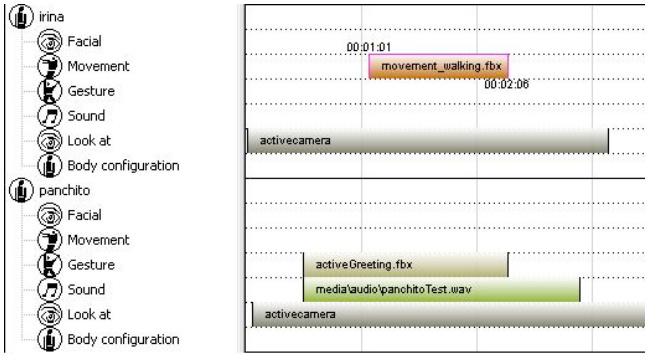

Figure 6: Tracks with two characters and several components added to the scene.

4.4 Output

The output of the Programme Editor follows the schema set out in Section 3.3 above. The real-time preview is a powerful feature of the editor as it loads the graphical engine in a sub-window to preview the programme. In the main window, a vertical red bar travels along the timeline to indicate exactly which section of the programme is currently being previewed. Standard video control buttons (Play, Pause, Stop, Forward, Reverse) allow control over the output, while clicking on the timeline will skip instantly to that moment in the programme. The viewer supports real-time addition of components to the programme: without pausing the action, a user can add a component to the scene and it will be displayed instantly. Figure 1 shows an example of the real-time preview window laid in front of the Programme Editor.

The other output option is the direct creation of a video clip, accessed through the menu bar of the main window. This is in fact a GUI for a separate command line programme that generates the video. The GUI provides a variety of options as the resolution and file types required for the video (as explained in Section 3.3) and allows the user to specify a name and location for the output video file. When creating videos with dynamic content, this final step is frequently unnecessary, as the videos will be generated separately with different content (as demonstrated in Section 5.1).

4.5 Formats and Technical Requirements

The output of the Programme Editor, and in fact the whole framework, is obviously highly dependent on the quality of the visual assets that are used. The file format supported by the framework is FBX, chosen as it is allows assets (objects, cameras, scenes, characters, animations) to be exported from several major art packages (such as both Autodesk Maya and 3D Studio Max). However FBX is still developing as a format and so we have developed guidelines for the best methods to develop and export assets. For use with the Programme Editor, assets must be placed in a directory which is specified by the user as the 'source' directory; all assets within here will be accessible through the context menus of the software.

The technical requirements for the Programme Editor vary according to the complexity of the scenes being designed and characters being used. However, minimum requirements for the real-time visualisation are generally a Windows-based machine with a GPU capable of running OpenGL Shader Model 2.

5. USE CASES

This section list several cases in which the automatic programme generation framework has been used. None of the use cases present a comprehensive evaluation of the technology, however the mix of commercial and research based applications demonstrates that the framework and applications have been used across a broad spectrum of applications.

5.1 Virtual Presenters

5.1.1 Virtual Weatherman

The most prominent use case of our framework is shown with "the world's first virtual weatherman", Sam [7]. Sam is a virtual character capable of presenting the weather for hundreds of locations around the world, and was created by Activa Multimedia Digital[1], in collaboration with our group and La Salle Engineering at Universitat Ramon Llull[2]. Figure 7 and Figure 8 show screenshots of Sam in action.

The development of Sam used the Programme Editor to create the block template, and then took advantage of the ability to modify the programme description separately according to external input data – in this case, the weather forecast. Three times a day, the latest weather 48-hour weather forecast for hundreds of locations is accessed from an online weather database. This data is then used to generate a video, using the visualisation technology presented in this paper, for each individual forecast for every location that is covered by the system. Multiple versions of these videos are then made available to download on a variety of platforms, including television and mobile platforms.

5.1.2 Virtual Stockmarket Report

A sample project in collaboration with a multinational media company demonstrates further use of the technology in the creation of programmes. The workflow for the virtual weatherman. Characters were designed and animated, while the Programme Editor was used to locate them within a scene. Up to date stock market information was used as text input for the character's speech, and videos generated periodically to present the information. A screenshot of one of these videos is presented in Figure 9.

Figure 7: Sam the virtual weatherman

[1] Activa Multimedia Digital. c/ Gasper Fàbregas 81, Barcelona 08950. http://www.activamultimedia.com/.

[2] La Salle, Universitat Ramon Llull, Claravall 1-3, Barcelona 08022. http://www.salle.url.edu/.

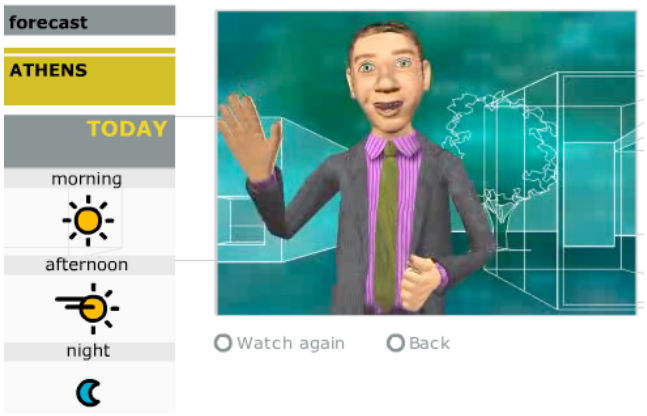

Figure 8: Sam explaining today's weather

Figure 9: Virtual stock market presenter

5.1.3 Virtual Bar-tender

"Manolo" is a virtual bar-tender who proffers opinions on the latest sports stories in a stereotypical manner. This use case differs from the previous virtual characters in that the audio comes is supplied by a recording of a real human voice (specifically from recordings of a sport-radio talk-show host). In that sense the technology is enabling a virtual view on non-visual events that have occurred previously. Figure 10 shows a screenshot of Manolo in action.

Figure 10: Manolo the virtual bar-tender

5.2 SALERO Project

SALERO (Semantic AudiovisuaL Entertainment Reusable Objects) [8] is an FP6 European Union Integrated Project which aims at making cross media-production for games, movies and broadcast faster, better and cheaper by combining computer graphics, language technology, semantic web technologies as well as content based search and retrieval. The technology presented in the paper has been the basis of a considerable amount of integration within the project.

5.2.1 Programme Generator Server

The automatic programme generation technology has been adapted for use with server technology, in a similar manner to Xtranormal [3]. Typical applications of this include the ability for users to specify different configurations within a scene (for example, choice of character, clothes, speech etc.) and then download a video of the generated scene (an Adobe Flash video in our implementation). Figure 11 shows a basic example. The limitations of what is possible with such a web-portal are due largely to the design of the interface page. It would be perfectly possible to create a comprehensive web-portal that mimicked the majority of the functions of the Programme Editor. Possible use case for this technology are in the ever growing field of user-generated content, allowing companies to let users create custom animations or videos according to their marketing plans.

5.2.2 Voice Generation and Transformation

Web based access to text-to-speech [9] and voice transformation [10] systems allows the creation of web-portals that combine these technologies into one application. Within the SALERO project such collaborations are resulting in proof-of-concept web-based mash-ups that show how such integration can rapidly create results (see Figure 11).

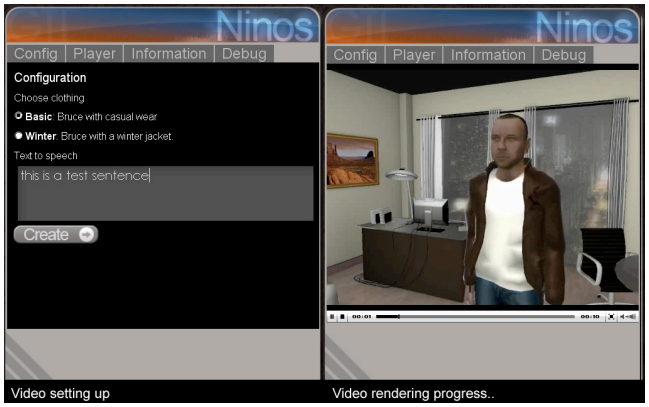

Figure 11: Programme Generator Server. On the left the user can specify some characteristics, including text to utter, on the right the resulting video is displayed

5.2.3 Automatic Gesture Suggestion from Audio

Speech tagging systems are able to extract time-correlated speech stress levels from a given input audio file. It is possible to use the output of such algorithms to drive automatic gesture suggestion for virtual characters within our framework. This cuts down on the time required for the user to create a programme description, adding a further layer of automation to generation process.

6. CONCLUSIONS AND FUTURE WORK

This paper has presented a framework for the assisted creation of computer-generated audiovisual productions. The use cases presented cover both commercial applications and more academic-based integrations. The commercial use cases demonstrate how the framework and applications are viable for use in a commercial environment, complete with the economy based time-pressures that are usual in such situations. The more academic use cases show that the framework forms a solid base upon which several interesting integrations and research collaborations can take place.

Yet further evaluation is required to paint a more accurate picture of the worth of the framework. While it is difficult to compare the framework directly with related work, it certainly is possible to perform user evaluation as to whether the framework and associated applications do genuinely facilitate the creation of audiovisual clips (although the existing commercial use seems to suggest already a positive answer to this question). This further evaluation will then be our immediate future task.

Further work lies in the correction and modification of several technical issues. A major improvement will be the introduction of animation blending within the same track. Animation blending is already supported for animations that lie on separate tracks (e.g. a walking animation can run concurrently, and is blended, with a waving gesture), yet the framework does not support animations overlapping on the same track. This means that any gesture, for example, must finish before a new one starts. Further ideas for improvements are the capability to use the visual preview screen to move objects with the scene, in a drag-and-drop manner. Yet going down this path may lead to several pitfalls, as 3D scene modification is something that is already covered comprehensively by several large modelling software programmes.

In conclusion, the framework and application presented in this paper presents a novel and powerful methodology for assisted animated production creation and programme generation. The heavy use that the framework has already seen suggests that it provides real benefit to production workflow, and provides a strong platform for the integration of multimedia technology.

7. ACKNOWLEDGEMENTS

The authors would like to thank the support and inspiration provided by Activa Multimedia Digital, Barcelona. Aspects of the work in this project were support by the European Commission FP7 IP SALERO[8], and the Spanish collaborative project i3Media.

8. REFERENCES

[1] Bulterman, D. and Rutledge, L. 2008. SMIL 3.0 - Interactive Multimedia for the Web, Mobile Devices and Daisy Talking Books. X.media.publishing. ISBN: 978-3-540-78546-0.

[2] Redboard. http://www.redboard.tv/.

[3] Xtranormal. http://www.xtranormal.com/.

[4] Girgensohn, A., Boreczky, J., Chio, P., Doherty, J., Foote, F., Golovchinsky, G., Uchihashi, S., Wilcox, L. 2008. A semi-automatic approach to home-video editing. Proc. ACM User interface software and technology (San Diego, USA, 2008). 81-89.

[5] Apple. “iMovie”. http://www.apple.com/imovie/.

[6] Adobe. “Premiere”. http://www.adobe.com/products/premiere/

[7] Meteo Sam. http://www.meteosam.com/.

[8] SALERO (Semantic AudiovisuaL Entertainment Reusable Objects). http://www.salero.info

[9] Alías, F., Iriondo, I., Formiga, L., Gonzalva, X., Monzo, C. and Sevillano, X. 2005. High quality Spanish restricted-domain TTS orientated to a weather forecase application. Proc. 9th European Conference on Speech Communication and Technology (Interspeech) (Lisbon, Portugal, 2005). 2573-2576.

[10] Mayor, O., Bonada, J. and Janer, J. 2009. Voice Transformation from interactive installations to Video-Games. Proc. 25th AES Conference: Audio for Games (London, UK, 2009).