Artificial Actor Behaviours for Interactive Videos: Handling AI Requirements in ELEVATE

Paolo Busetta, Maurizio Astegher, Daniele Dellagiacoma, Stefano Gabrielli, Matteo Longato and Matteo Pedrotti
DeltaLab Research and Development Division
Delta Informatica SpA, Trento, Italy
{paolo.busetta, maurizio.astegher, daniele.dellagiacoma, stefano.gabrielli, matteo.longato, matteo.pedrotti}@deltainformatica.eu

Abstract

Animating artificial character on a virtual stage requires an exacting description of all details of the movements of the body and in space and of the actions to be performed, in particular when plausible behaviours are requested. Behavioural AI, i.e. a character animation machinery that automatically adapts to context, helps in reducing efforts but is difficult to develop. This paper discusses a production process for interactive training videos that allows three stakeholders – an instructional designer defining an exercise; a director shooting video clips from virtual reality with artificial characters; and, a development team creating models of behaviour – to efficiently cooperate by means of nested requirement / implementation / testing cycles. Good requirements and appropriate AI technologies (typically a mixture of goal-oriented programming and statistical models) allow the creation of behavioural models that can be used for many exercises, thus greatly reducing costs compared to *ad hoc* scripting.

1 Introduction

A brief description of the desired behaviour is usually more than enough for actors on a stage; human intelligence, with its understanding of the context and of the impersonated characters, fills the gaps. By contrast, directing an artificial character on a virtual stage requires an exacting description of all details, in particular when plausible behaviours are requested according to a very precise script that cannot simply be forced onto a statistical model or learned somehow from existing data collections. The resulting implementation cost is one of the reasons that limit the application of virtual reality (VR) in training, even if VR could allow human participants to improve their skills in dangerous and rare situations, e.g. emergency management with victims and crowd involved.

Models of procedural behaviour such as behaviour trees [CÖ17] and goal-oriented programming [Ork03] that adapt to the context in which artificial characters act (called *behavioural AI* from now on for brevity) could

In: M. Sabetzadeh, A. Vogelsang, S. Abualhaija, M. Borg, F. Dalpiaz, M. Daneva, N. Fernández, X. Franch, D. Fucci, V. Gervasi, E. Groen, R. Guizzardi, A. Herrmann, J. Horkoff, L. Mich, A. Perini, A. Susi (eds.): Joint Proceedings of REFSQ-2020 Workshops, Doctoral Symposium, Live Studies Track, and Poster Track, Pisa, Italy, 24-03-2020, published at http://ceur-ws.org

potentially alleviate this situation if reused many times. To this end, good requirements, capturing the right abstractions that allow reuse in multiple situations and domains, are essential.

This paper introduces ELEVATE, a tool suite for the production of interactive videos for online training, and its virtual stage for video shooting, called EVE. We discuss how behavioural AI models are incrementally added to EVE, thus progressively enriching the tool suite itself, by means of a process that facilitates the dialogue among domain experts, content creators and platform developers.

Next section presents the ELEVATE application domain and its main components. Section 3 briefly reviews how AI is currently applied to the production of videos and in video games. Section 4 discusses the process supported by ELEVATE to allow domain expert to describe their requirements and drive the development team in building behavioural models. A few final remarks conclude this paper.

2 Context: Interactive Videos as Exercises for Learning Management Systems

With the availability of high-speed Internet and the pressure for continuous training, online e-learning has become very popular, especially in professional fields requiring mandatory skill certifications and facing continuous flows of business, legal and technical novelties. A number of so-called Learning Management Systems (LMS), both proprietary and open-source (e.g. Moodle[1]), are available to deliver courses created with content-generation tools, whose current market leader is Articulate[2].

In self-paced courses, with no interaction (not even indirect) with human trainers, videos are commonly used to deliver theory and examples, complementing textual contents. A variety of multimedia interactive quizzes and tests allow students to check their understanding and possibly undergo an examination; they range from multiple-choice questionnaires to game-like interactions and simple simulations. Virtual reality (VR) is a promising technology for these purposes, with supporting devices readily available thanks to their popularity in video-gaming. However, even today the cost of development and limited user acceptance outside of the youngest audience makes VR applied almost exclusively to the most complex training and with human supervision.

A cheaper alternative to full VR, but typically leading to higher levels of engagement than traditional linear videos and questionnaires, is the use of interactive videos in which the trainees have to take decisions in order to progress through different sections of a filmed story. This may even have different endings, some positive and some negative according to the choices taken, so it may be used to test competences. The scripting of an interactive story for training purposes, the shooting and editing of videos or, equivalently, the preparation of animated multimedia material are non-trivial tasks that require specialised tools and an instructional designer with both technical and educational skills.

2.1 ELEVATE: Script Editor and Virtual Stage

The ELEVATE research project has produced a tool suite for the creation of interactive videos to be used as exercises in online courses. Differently from other tools, where an instructional designer acts also as multimedia creator, ELEVATE clearly distinguishes two different user roles: the *exercise designer*, which typically is a domain expert or an instructional designer that defines training objectives and how to verify them, and the *director*, who produces content according to the requests by the exercise designer.

The first ELEVATE tool of interest for this paper is a graphical story editor for the exercise designer, used to define the structure of an exercise. The latter is a directed, cyclic graph whose nodes ("fragments") are placeholders for video clips; the edges outgoing from a node ("choices") represent options that are offered to a trainee at that point in the story, possibly according to contextual parameters. The destination of an edge is the point from which the story will resume if that edge is followed, i.e. if the trainee takes that choice. Thus, a path through a story graph represents a view of the video or, equivalently, a sequence of choices.

Before designing a story structure, the exercise designer needs to pinpoint the goals of an exercise and the criteria used to judge achievement or failure. Goals, input parameters, trainee decisions and other run-time context elements are represented with a simplified logical language ("tags").

Once goals have been defined and the exercise structure designed, the exercise designer must link fragments to video clips so that the complete interactive video can be generated. The clip for a fragment may be loaded from an existing media library or recorded *ad hoc*. In this case, the exercise designer notifies the director through a so-called *video recording request*, which links back to the exercise and to the specific fragment to be satisfied.

[1] https://moodle.org/
[2] https://articulate.com/

The director is in charge of deciding on secondary or under-specified aspects (e.g. camera perspective, objects on the scene) and for recording.

Videos can be recorded either externally, e.g. by shooting in real-life settings or exported from a multimedia authoring system, or by means of another tool of the suite called EVE (ELEVATE Video Editor). This is a VR system developed with the popular Unity game engine[3]. EVE offers the director a virtual environment where to create situations that could be difficult to be replicated in the real world for many reasons, e.g., a dangerous fire emergency involving a crowd and many objects, or privacy restrictions on real-life shooting. EVE allows the director to load a 3D scenario and populate it with a variety of artificial actors (so-called NPCs, Non-Player Characters) and graphical assets. The director specifies a simple script for the NPCs to follow in order to satisfy a video recording request. Typically, a script is a sequence of goals to be achieved by the underlying goal-oriented multi-agent framework that animates the NPCs. Once the script is ready, the director records the scene, letting the NPCs to behave autonomously in the virtual environment as if they were actors on a real scene. EVE allows the director to save the state of the virtual stage at any time ("checkpointing"), from which to restart to satisfy the video recording request of the following fragment.

Once ready, the recorded video is uploaded to the media library by the director and the video recording request closed. The exercise designer either approves the video or rejects it, reopening the recording request with comments explaining the reason for the refusal. Conversely, the director can provide feedback to the exercise designer before recording; furthermore, if a request cannot be satisfied for any reason related to the exercise, e.g. the instructions on the fragment are not clear, the request is put into an "error" state. It is left to the exercise designer to change the request as appropriate and resubmit it to the director.

The video recording request cycle just described is similar to those of tickets in popular issue trackers, e.g. Bugzilla. This cycle may be extended in various directions. For instance, feedback from the director may lead the exercise designer to rethink the exercise and readjust the story graph. Conversely, the director may discover that the available capabilities are inadequate; most importantly for the following discussion, behaviours of NPCs may be missing or inappropriate and consequently trigger an inner cycle of development involving an AI developer.

When all video clips for an exercise have been loaded and approved, the exercise designer generates the complete interactive video, which is composed of a standard multimedia file and of ELEVATE-specific metadata containing structure, goals and other parameters as mentioned above. An ELEVATE player, compatible with any Web-based LMS supporting the SCORM standard (part of the IEEE 1484 series for eLearning Technologies), allows a trainee to do the exercise and keeps track of its progress.

3 AI Applied to Video Production and Video Gaming

AI has only recently seen an actual improvement and utilisation in film making. As the technology progresses and the use of machine learning-based approaches such as neural networks is growing in industry, many roles have become (or are becoming) ready for automation. One example is script-writing AI [Dal18], which gives a glimpse of what, in the future, AI could do (even if today the results are still pretty awkward and limited). Another example is the camera-manning role [Dal18] which aids the director in shooting complex sequences that may involve camera-carrying drones and may simplify the post-production work of mixing and editing. However, so far the complexity of the visual storytelling (including the ability to convey emotions and narrate a story relying on images rather than precise text describing every detail) has prevented the full automation of creation and understanding of complex editing or compositions. In recent times, AI is being trained and exploited to further assist directors, with research topics such as shot labelling and camera movement recognition [Som19]. Furthermore, AI is being trained to assist 3D artists, learning how 3D modelling and 3D animations are made [Fou19, Smi16].

Of particular relevance for ELEVATE, AI is commonly used in video games and, by extension, in serious games and VR environments based on game engine technologies. It is necessary to clarify that, in video gaming, traditionally the term AI does not refer to machine learning or symbolic reasoning but to anything, including well-known algorithms and design patterns, that allows game developers to simulate real-life behaviour; the final objective, of course, is to give the player the impression of dealing with intelligent entities. Until recently, game AI would not rely on any complex machinery because they require more resources than those available on a player's computer. Because of these limitations, often the AI used in video games is nothing more than a set of (sometimes complex) behaviour trees [CÖ17] driving the actions of NPCs, meaning that no real reasoning is involved and NPCs show little or no sensitivity to the evolution of the situation [SML19]. As a consequence,

[3] https://unity.com/

creating a plausible, complex behaviour becomes challenging, with the lurking risk of sacrificing the player experience because of the poor AI [SML19]. To overcome this problem, many developers have chosen to give the AI some kind of advantage with respect to the human player: for instance, to simplify its behaviour tree, an AI agent can directly access some non-observable information (such as the player position in the game at every instant, even when in hiding) rather than purely relying on a realistic perception of the situation from the perspective of the NPC. This approach, commonly called "cheating AI", can be perceived as an unfair advantage when overdone, e.g. by making it impossible to apply any effective counter-tactics when the user is competing against NPCs, eventually leading to user frustration [Sco02].

Some more modern AI patterns and approaches are being adopted with the evolution of the consumer's hardware and of the industry itself, which is subject to strong competitive pressure and thus prone to sacrifice realism to meet deadlines. A well-studied example is the GOAP (Goal-Oriented Action Planning) system [Ork03] that, even if it has been developed in the early 2000s, it is still not so widely used because of the higher resource need, compared to standard behaviour trees [Ave18].

We discussed in previous works [DGB+15] how to define semantics for a VR so that NPCs (or, more precisely, the software agents that control the NPC behaviours; typically, one agent per NPC with a classical multi-agent approach) can autonomously work out context-specific details. These include, for instance, the classification of locations and objects. In [BD15] we presented a technique for goal-oriented model development based on the BDI model of agency, in principle similar to the GOAP system.

Some types of game AI deal with aspects other than NPC behaviours, for instance, story generation, procedural environment generation and so on, but their discussion is outside the scope of this paper.

4 Designing Controllable AI Models for Virtual Actors

This section focuses on the process supported by EVE for recording a video as a request by an instructional designer and, in turn, how this may generate requirements for model development. We illustrate this process with an example taken from emergency training, where firefighters are involved in rescuing the victim of a car accident; the trainee has to choose which procedure has to be applied by the emergency team.

4.1 AI in EVE

The purpose of behavioural AI in EVE is to reduce the workload on the director by providing reusable "intelligent" models that minimise the number of *ad hoc* animations and scripts that are normally required for the development of a so-called "game level", i.e. of a scene and its actors in a complex video game. The director transforms high-level requirements by the exercise designer (typically expressed in natural language) into scenario settings, directives for the existing behavioural AI and, when necessary, in additions or improvements of the latter demanded to an AI development team.

The fact that there is no game logic in EVE - there is no player during video recording, indeed - is practically irrelevant in terms of the inherent complexity of the graphical and behavioural models of the NPCs / virtual actors. This complexity is very variable, of course. On one extreme, shooting a static scene, e.g. an introduction to a story that allows the trainee to familiarise with a certain location such as a street corner with sparse road traffic, is a matter of choosing a 3D environment, populating it with objects that at most execute loops of stereotyped animations, deciding on the camera position – that is, basic editing activities that require minimal skills by the director and little "intelligence" in the game engine and its models. On the opposite extreme, a very animated scene with characters performing complex and coordinated routines, e.g. two fire-fighters that extract the screaming victim of an accident out of a car on fire, requires a substantial amount of work that, if done purely by means of a 3D editor or a basic game development environment such as Unity, could be time-consuming if not challenging even for an expert multimedia specialist.

EVE is engineered in such a way that 3D scenarios, 3D object models with their basic procedural animations, and behavioural models written in a goal-oriented fashion can be imported from libraries rather than being built-in. The director chooses what fits her purposes and writes relatively simple scripts that submit goals to NPCs in a sequence as well as some minimal event-based coordination (following a pattern such as "give goal G1 to character A1 when event E1 has happened" – e.g. "give goal VICTIM V1 EXTRACTED to FIREFIGHTER F1 when DOOR D1 OPEN").

As mentioned, behavioural models are created by a specialised AI team according to the needs of the director and exploit all available geometrical data and semantic annotations to "intelligently" adapt to different contexts. Models for EVE can freely "cheat" without worrying about the user experience because agents are not competing

against a player. So, unlike the standard GOAP definition of world knowledge [Ork04], EVE agents can access every piece of information they need to reach their goals. simplifying the models; further, the manipulation of information shared by all agents and maintained by the game engine allows for complex coordinated behaviour.

That notwithstanding, previous experience with model development has shown that it is critical to reduce misunderstandings by the modeller / developer to prevent both frustration in VR designers and disappointment of final users. The engineers developing models for a new domain of application of EVE (e.g. fire-fighting in buildings is different from open street accidents) cannot know all models required to satisfy all exercises in advance, so they need to work closely with the director until a sufficiently rich model library is available to let her work without further support. To this end, a short development cycle with user involvement, such as the video recording request management in ELEVATE described below, is an important enabling precondition.

4.2 The EVE Requirement-to-Model Cycle for AI

The interaction between the director and the EVE model developers follows a classic requirement specification / development / testing / user acceptance cycle, currently supported by a standard issue tracker. Internally, the EVE model development team may adopt any software engineering methodology they deem appropriate. We illustrate this process with an example of video refinement causing the creation of new models:

1. The exercise designer submits the video recording request for a fragment described as "the firefighters extract the victim from the car then one checks her conditions while the other puts out the fire".

2. Since this is not the first shot of the exercise, the director starts from the final checkpoint of the fragment that leads to this one, otherwise she would have to set the scene up from scratch. As mentioned, it includes two firefighters (F1 and F2) and a victim in a burning car. The director writes the script, whose pseudo-code is: F1 is given goal "driver door opened"; when opened, F2 is given "victim moved to the ground"; when the victim is on the ground, F1 is given "extinguisher opened towards fire" and in parallel F2 is given "three agitated arm movements directed to victim performed". The video is recorded, stopping when the director considers that it is long enough, then uploaded to the media library and the video recording request closed.

3. The exercise director rejects the video, reopening the request, because the animation corresponding to checking the health condition is not plausible: the firefighter must recline above the victim and check breathing and heartbeat in this order, finding no problem.

4. The director analyses the designer's comment and, in turn, requires the model development team to create three different models for the firefighter satisfying these goals: reclined to a free side of character C; breathing of character C checked; hearth beat of C checked. This is done by creating three entries in the issue tracking system of the AI development team; each issue includes, as accompanying documents, a read-only link to the video recording requests, thus all exchanges between exercise designer and director as well as the rejected video are available to the developers. Conversely, the director adds links to the created issues to the video recording request itself, as comments for the exercise designer; while this step is not strictly necessary, it gives an opportunity for the designer to be involved in addition to justifying a delay in production.

5. Developers produce the required models and upload them in the model library. User acceptance is performed by the director that invokes the new models from its script and evaluates the results. The model development cycle is repeated until the director is satisfied.

6. The video production cycle restarts from step 2 above and is repeated until the exercise designer approves the result.

A detailed discussion of model creation by the development team would be outside of the scope of this paper. In short, with the goal-oriented agent framework currently adopted for EVE (derived from [BRCG17, BCR16]), it consists of the definition of a set of accepted goals, with their parameters, and the development of context-sensitive plans to achieve them, plus the semantic annotations required on the environment and its objects to understand the context [DGB$^+$15, BD15]. The process described above would equally apply to other AI technologies or even simple *ad hoc* scripting, with unavoidable limitations on the reusability of models.

4.3 Discussion

In the EVE video production process, the behavioural model development team receives clear inputs concerning execution context and desired behaviours as well as a specification of the acceptance tests (step 5): they are nothing else than the fragment to be recorded and the overall exercise structure, plus the breakdown in goals defined by the director in step 4. They reduce the need to create other typical RE artefacts, such as functional specifications and test cases.

The nested cycle presented above differs from typical serious game, simulation and multimedia development processes because exercise designer, director, and developers work separately and at different times rather than as a single team focusing on the next product release. Developers aim at creating a framework that enables the first two to generate an arbitrary number of videos by themselves. Further, they may be part of different organisations. Indeed, in the current ELEVATE use cases, designer and director are part of the training office of a large public department, while the development team is a private company. The business objective of the latter is to make its products as reusable as possible, thus targeting a community of users creating training material for different audiences and possibly a marketplace of models.

ELEVATE's feedback-oriented iterative cycle presents some analogies with those supported by tools for requirements collection directly from usage context, for instance the ConTexter proposed by Wehrmaker et al. [WGS12] and the Mobile Scenario Presenter (MSP) of Seyff et al. [SGGM07]. These systems focus on software-intensive systems in real-life settings, and capture feedback from the user while using a system. In the case of EVE, which specifically focuses on virtual reality applications, context is established by the users themselves (exercise designer and director) and captured by the video recording request.

5 Conclusion

We have introduced the ELEVATE tool suite for interactive video production. One of its components, EVE (ELEVATE Video Editor) uses AI to animate virtual actors on a virtual stage. In particular, EVE uses behavioural AI, i.e. symbolic reasoning or some form of machine-learned logic implementing context-sensitive procedures, as a better alternative to hard-wiring simple algorithms or case-by-case scripting as done by most commercial entertainment and serious games. The objective is to reduce the costs of production of videos by supporting a high level of adaptability of its virtual actors to different situations.

While the idea of reusable AI models, shared by a community, is sound in principle, our own experience is that good models are very difficult to develop because it is hard to capture truly context-independent requirements in a systematic way, over time and with the continuous involvement of domain experts. The presented approach aims at alleviating issues by streamlining the process of requirement capture and implementation and by separating content conception and production from platform development, thus enabling incremental improvements over time that can be shared by a community of users.

All base components of EVE have been developed and used by previous projects or are taken off-the-shelf. However, the complete interactive environment is still under development at the time of writing. Experimentation on the full EVE is expected to happen by Spring 2020.

Acknowledgements

This work is part of the ELEVATE research project, which is funded by Provincia Autonoma di Trento, L.P. 6/1999.

References

[Ave18] D. Aversa. Choosing between behavior tree and goap (planning). `https://www.davideaversa.it/2018/02/choosing-behavior-tree-goap-planning/`, 2018. [Online; accessed 19-December-2019].

[BCR16] P. Busetta, P. Calanca, and M. Robol. Applying bdi to serious games: The presto experience. Technical Report DISI-16-011, Version 2.0, University of Trento, Italy, 2016. http://hdl.handle.net/11572/154697.

[BD15] P. Busetta and M. Dragoni. Composing cognitive agents from behavioural models in presto. In *16th Workshop "From Objects to Agents", Naples, Italy*, June 2015.

[BRCG17] P. Busetta, M. Robol, P. Calanca, and P. Giorgini. Presto script: scripting for serious games. In *AI & Games Symposium at AISB, University of Bath (UK)*, 2017.

[CÖ17] M. Colledanchise and P. Ögren. Behavior trees in robotics and ai: An introduction. *CoRR*, abs/1709.00084, 2017.

[Dal18] D. Dallas. Filmmaking, artificial intelligence & machine learning. `https://creativefuture.co/artificial-intelligence-automation-film-video-machine-learning-editing`, 2018. [Online; accessed 19-December-2019].

[DGB+15] M. Dragoni, C. Ghidini, P. Busetta, M. Fruet, and M. Pedrotti. Using ontologies for modeling virtual reality scenarios. In *2th European Semantic Web Conference, ESWC 2015, LNCS, volume 9088*, pages 575–590. Springer, 2015.

[Fou19] Myelin Foundry. Deep learning can democratize animation and vfx. `https://becominghuman.ai/deep-learning-can-democratize-animation-and-vfx-43bf51f06090`, 2019. [Online; accessed 19-December-2019].

[Ork03] J. Orkin. Applying goal-oriented action planning to games. *AI game programming wisdom*, 2:217–228, 2003.

[Ork04] J. Orkin. Symbolic representation of game world state: Toward real-time planning in games. In *Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence*, volume 5, pages 26–30, 2004.

[Sco02] B. Scott. The illusion of intelligence. In *Rabin, Steve (ed.). AI Game Programming Wisdom. Charles River Media.*, pages 16—20, 2002.

[SGGM07] N. Seyff, F. Graf, P. Grünbacher, and N. Maiden. The mobile scenario presenter: A tool for in situ requirements discovery with scenarios. In *15th IEEE International Requirements Engineering Conference (RE)*, pages 365–366. IEEE, 2007.

[Smi16] J. R. Smith. Ibm research takes watson to hollywood with the first "cognitive movie trailer". `https://www.ibm.com/blogs/think/2016/08/cognitive-movie-trailer/`, 2016. [Online; accessed 19-December-2019].

[SML19] L. E. Shummon Maass and A. Luc. Artificial intelligence in video games. `https://towardsdatascience.com/artificial-intelligence-in-video-games-3e2566d59c22`, 2019. [Online; accessed 19-December-2019].

[Som19] R. Somani. Ai for filmmaking. `https://rsomani95.github.io/ai-film-1.html`, 2019. [Online; accessed 19-December-2019].

[WGS12] T. Wehrmaker, S. Gärtner, and K. Schneider. Contexter feedback system. In *34th International Conference on Software Engineering (ICSE)*, pages 1459–1460. IEEE, 2012.