A Virtual Agent Toolkit for Serious Games Developers

Samuel Mascarenhas, Manuel Guimarães, Rui Prada, João Dias and Pedro A. Santos
INESC-ID and Instituto Superior Técnico
Universidade de Lisboa
2744-016 Porto Salvo, Portugal
samuel.mascarenhas@gaips.inesc-id.pt,
manuel.m.guimaraes@ist.utl.pt,
rui.prada@tecnico.ulisboa.pt,
pedro.santos@tecnico.ulisboa.pt,
joao.dias@tecnico.ulisboa.pt

Kam Star, Ben Hirsh
and Ellis Spice
PlayGen
8-9 Talbot Court, London, UK
kam@playgen.com,
ben@playgen.com,
ellis@playgen.com

Rob Kommeren
Stichting Praktijkleren
3821 AR Amersfoort, NL
r.kommeren@stichtingpraktijkleren.nl

Abstract—The design of serious games requires developers to tackle pedagogical challenges calling for advanced solutions that the entertainment industry might deem too risky to pursue. One such challenge is the creation of autonomous socially intelligent characters with whom players can practice different social skills. Although there are several architectures in the field of virtual agents that are designed specifically to enable more human-like interactions, they are still not widely adopted by game studios that develop serious games, in particular for learning. In this paper, we present a virtual agent toolkit that was specifically developed with the intent of making agent-based solutions more accessible and reliable to game developers. To this end, a collaborative effort was established with a game studio that has used the toolkit to develop two different serious games. Among other advantages, the toolkit facilitated the inclusion of a dynamic model of emotions that affects not just how the character looks and acts but also how the player’s performance is determined.

Index Terms—Serious Games, Virtual Agents, Authoring Tools, Interactive Storytelling, Affective Computing

I. INTRODUCTION

The industry of video games has seen tremendous growth to the point that the budget for highly anticipated games can surpass the cost of big Hollywood films [7]. This led to extensive development times and quite large development teams and corresponding high expectations from the players [13]. On one hand, this state of affairs has enabled the creation of very detailed game worlds with stories and characters that players find very engaging to interact with. But, on the other hand, the huge risk that is now associated with failing to meet the expectations of players has led the industry to primarily focus on what has been known to work in the past. This is also then reflected in the available development tools, with popular game engines like Unity\(^1\) being primarily designed to support the typical requirements and methods used in entertainment games that were previously successful. As a result, game developers that are interested in developing games with more unique characteristics or requirements, which is often the case for pedagogical games, usually find themselves having to spend a significant amount of time in developing their own tools and methods.

The serious games industry is growing as well, supported by the continuous research on the potential in using games for other purposes than just entertainment [10], [12], [21]. Serious games can be used to train and teach players on various subjects (e.g. math fractions [14], logic operators [16]) or raise awareness on social issues (e.g. sustainability [17], cultural diversity [4], bullying [20]). In fact, one of the more interesting aspects in developing games that are designed to teach is that their design is centered around pedagogical challenges. As such, even if the game is very engaging for players it will still fail to achieve its purpose if it does not have a pedagogical outcome. But, in turn, the game might have great pedagogical content but fail to deliver it in an engaging manner. One of the important aspects that make players engaged in a game world is the appeal of its characters. Particularly, non-player characters provide the opportunity for the player to engage in social interactions in a safe environment and within the confines of the game rules and structures. From a training perspective, players are free to experiment and observe the effects their actions have on simulated others in order to obtain and practice certain social skills. However, the range of social interactions that are typically offered to players is still quite limited when compared to real human interaction.

With the goal of expanding the range and complexity of social interactions between characters and humans, there has been a substantial amount of research dedicated to the creation and study of virtual agents. These are embodied characters that are designed to be able to interact with humans in a natural manner [8]. The architectures that have been developed for these characters can be rather complex, having to deal with the challenges of interpreting and synthesizing both verbal and non-verbal actions as well as modeling cognitive and affective processes related to decision making.

Although researchers have been able to successfully apply

\(^1\)https://unity3d.com/
virtual agent architectures in the development of serious games (e.g. [1], [9], [11]), such architectures have not yet been widely adopted by game studios. While the accessibility of these architectures can be improved through the creation of better graphical user interfaces and more extensive documentation, there are also technical and conceptual issues that must be addressed [18]. A virtual agent architecture relies on a type of authoring that is oriented towards cognitive concepts such as goals and beliefs, which are quite familiar for AI researchers but not necessarily so for game developers. Also, an agent model will promote a type of storytelling experience that is distributed or character-centric [2] whereas popular game developer tools like Articy:draft\(^2\) or Twine\(^3\) are designed towards a plot-centric approach with branching dialogues. While these tools can be used to create complex narratives they make a strong distinction between the player and the other characters, by giving dialogue options to the former but not the latter. In the proposed toolkit, while certainly possible, it is not necessary to tie dialogue options to a specific character or the player.

In this paper, we present a novel toolkit that aims to promote the adoption by game developers of virtual agent tools for creating game characters that are more socially and emotionally intelligent (e.g. are able to adapt to the situation and to the players). The toolkit is based on the existing FAtiMA Modular architecture [5], which is an architecture that was has been successfully used in the past in several research applications [1], [3], [4]. These improvements were derived from a close collaboration with game developers at the company PlayGen\(^4\) that used the toolkit to develop two games for learning. The first one is named Space Modules Inc and is being developed for a European educational institute in the Netherlands named Stichting Praktijkleren\(^5\). The game is designed to teach its players how to provide better customer service in technical support. The second game is named OKKAM\(^6\), a spinoff company of the University of Trento in Italy. It is a single player game where players assume the role of a sailing team manager. Players must hire, fire and communicate with their team members in order to succeed and, therefore, learn some personal management skills.

This collaboration is part of the ongoing RAGE project\(^7\), which is an EU-funded project with the goal of developing and promoting new technologies for directly supporting applied game developers at creating better applied games and in a manner that is more cost-effective [19].

II. FATIMA TOOLKIT

FAtiMA Toolkit is an open-source project\(^8\) that contains a collection of tools and libraries with the aim of enabling the creation of interactive storytelling scenarios with non-player characters that can interact socially with human players in a variety of contexts.

Storytelling can bring multiple benefits to serious games [15]. Not only are people more likely to remember what they learned if the content is integrated in the context of a narrative, but also, an emotionally engaging story will greatly motivate players to achieve the intended learning goals of the game. This form of storytelling centers on the ability of players to shape how the story unfolds according to their actions, as participants rather than as observers. This feeling of agency increases player engagement and encourages them to reflect more deeply on the consequences of their choices. However, the more freedom given to players, the more difficult it becomes to use a traditional scripting approach to author the scenarios. This is because the branching factor of possible narrative paths quickly becomes intractable.

Our proposed storytelling framework deals with this issue by following a character-centered approach rather than a plot-centered one. The authoring is thus focused around the different roles that the characters might play in the game and the narrative emerges from how the characters behave in their given roles. The challenge then becomes to author these roles in a way that characters act in a believable manner but also serve the intended learning goals of the scenario.

As previously mentioned, the toolkit is the result of several improvements that were made to the FAtiMA Modular architecture [5]. For example, the code was ported from the Java language to C# in order to streamline the integration with game engines, such as Unity3D. Also, each component within the toolkit is able to fully load and save its internal state to a JSON file. As such, it is possible for the game developer to use his or her text editor of choice to do any kind of authoring task. However, the toolkit contains some complex data structures that refer to one another, such as emotions, an autobiographical memory, appraisal rules, among others. For this reason, each component has an authoring tool with a graphical user interface that help users’ in the creation of content in a declarative way preventing syntactical errors. The fact that the entire internal state of each component within the toolkit can be written to a file also works as a logging mechanism.

Many agent-based tools are designed to function as a framework or as a stand-alone application that the game must communicate with, using a specific protocol. In both of these cases, the game developer has to accommodate the game to how the agent tool specifies its communication protocol, its execution cycle and its extensions points, instead of the other way around. Moreover, given their opinionated nature, agent-based frameworks are difficult or even impossible to compose together. It was based on these limitations that we applied a functional library design pattern in the development of the toolkit. Consequentially, all the different components were developed as libraries, i.e. a collection of functions with well defined inputs and outputs, that the game developer can directly import and explore more easily without having to

---

\(^2\)https://www.nevigo.com/en/articydraft

\(^3\)http://twinery.org

\(^4\)http://playgen.com/

\(^5\)http://www.stichtingpraktijkleren.nl

\(^6\)http://www.okkam.it/

\(^7\)http://rageproject.eu

\(^8\)https://github.com/GAIPS-INESC-ID/FAtiMA-Toolkit
The main functionality of FAtiMA Toolkit is divided in two main components, the Role-Play Character and the Integrated Authoring Tool.

A. Role-Play Character

The Role-Play Character (RPC) is the name given to the component (see Figure 1) within the toolkit that manages each character’s reasoning and emotional state based on a perception-action mechanism, which can be described in the following manner. Firstly, the events that occur in the game world are sent as input to the Emotional Appraisal component, which is based on a formalization of the OCC cognitive theory of emotions [6]. This component then determines if the event will trigger a new emotion for the character. Each character can be configured with different appraisal rules that will result in having different emotional outcomes for the same events. After the emotional appraisal process is done, any resulting emotion is added to the Emotional State. Events are also stored in the character’s Autobiographical Memory along with any emotion associated to them. The character’s Knowledge Base keeps track of what the character believes as logical predicates such as $\text{Weather(Outside)} = \text{Raining}$. These beliefs are also updated according to the events sent by the game world.

After all the internal structures are updated, the RPC uses the Emotional Decision Making component to select the next action of the character. This is done using a rule-based mechanism that considers both the beliefs of the character as well as its emotional state. In addition to regular beliefs that are directly stored in the Knowledge Base, the decision-making process also takes into account meta-beliefs, which are added by Reasoning Components such as the Dialogue Manager or the MCTS. Syntactically, meta-beliefs are expressed in the same manner as regular ones. The key distinction is that, rather than being stored, the values of these beliefs is determined dynamically by the algorithm specified in the reasoning component. This allows the combination of multiple decision-making strategies into a unified rule-based system. Developers can also register their own modules as additional reasoning components and the meta-beliefs they introduce will become available in the conditional rules of all other components. For instance, consider a game with a specific scoring mechanism for the player and the developer wants to create a decision rule for NPCs to congratulate the player whenever the player’s score reaches a certain threshold. This could be achieved by registering the scoring mechanism as a new Reasoning Component that would add $\text{Score(Player)} = [x]$ as a new meta-belief.

Game characters should have believable emotional responses to give the illusion of life. For applied games that rely heavily on social interaction, it quickly becomes impractical to manually script all the emotional reactions of each character for each possible event. The RPC asset tackles this issue by allowing game developers to create general profiles of how characters respond emotionally in their games. They can test and configure these profiles outside of the game and they can naturally switch between profiles without having to recompile the game source code.

B. Integrated Authoring Tool

The Integrated Authoring Tool is the other main component of the toolkit that is designed to be the central hub for game developers when creating a new storytelling scenario or adapting existing ones. It allows the configuration of the general aspects of the scenario and provides quick access to the authoring tools of the Role-Play Character component. However, the main feature of this component is that it contains a dialogue editor that allows the developer to specify the dialogue acts that are available for both the player and the characters.

For the purpose of dialogue management, the author must define the interaction state where each dialogue may occur as well as define the next state if a certain dialogue is selected. During runtime, all characters are informed about the existing dialogue acts as well as dialogue states. Characters are then able to use this information to decide what to say according to their internal state and decision-making mechanisms. To give an example, consider that the integrated authoring tool informs a character that at the start of the interaction there are two valid dialogues, one to greet the player respectfully, another to greet the player in an angry manner. If the character is angry, the emotional decision making asset will select the second option. If not, then the first greeting will be selected instead.

III. CASE STUDY 1 - SPACE MODULES INC

Space Modules Inc is a single player game where the player takes on the role of a customer service representative for a spaceship part manufacturer “Space Modules Inc”. The virtual characters in the game play the role of customers that call the player (see Figure 2) about hardware and software faults they are experiencing. Some characters will be angry, others uncooperative or stressed, and it’s up to the player to manage the situation and decide how best to respond.

Players have to respond to situations by engaging in conversation with customers. This is done by having the player pick one of the available dialogue options in response to the character’s chosen dialogue. The process is repeated until the
The final state of the conversation is reached and then the player’s score is passed to the review screen to be shown to the player (see Figure 3). The customer satisfaction score depends on how the player affected the emotional state of the character. The idea is that each customer can have a different emotional profile, thus providing a different challenge to the player. From a pedagogical perspective, players must learn how to manage intense emotions and how to respond to customers in a professional manner in the best way. In other words, the pedagogical goal of the game is to train players in being able to identify a person’s emotional state through verbal and nonverbal feedback and gain further experience in providing effective emotional responses.

The emotional reactions of the customers in Space Modules are determined by the Role-Play Character component. According to the selected emotional profile, this component initializes the overall mood of the character to a given value between -10 to 10. The component then updates this value based on how it evaluates the option selected by the player. If the player decides, for instance, to give the wrong solution for the problem that the customer has, the component will generate a “Distress” emotion and the overall mood decreases. The player can then repair the mood of the character by selecting a dialogue that shows empathy for the character’s distress.

However, if this dialogue is selected when the character is not feeling distressed, then it will be judged negatively instead and the mood of the character decreases accordingly. The amount by which the mood decreases or increases is also another parameter that is possible to configure in the RPC component.

IV. CASE STUDY 2 - SPORTS TEAM MANAGER

Sports Team Manager is an applied game also developed by PlayGen with the assistance of the FAtiMA Toolkit. The overall goal of the game is to have the player be able to assemble together the most optimally performing sailing team by resolving conflicts and managing the team’s interactions. The player interviews virtual characters to identify their skills and personalities. The team has a set of roles, each with overlapping skill requirements. A successful sailing team is not solely based on skill, but also on the social relationships between team members. Players must communicate with their team, deciding which members are placed into each position per race and resolve conflict situations as they arise. Figure 4 shows the game flow during an individual race session.

The players must first review the positions they need to fill on the boat, taking note of the required skills for each. Next, they must meet with their NPC team members, taking into account the skills and inter-team relationships already known, asking questions where further information is needed. Using this information they should, if required, recruit new members into the team and place individuals into positions. After racing with the selected line-up, players will occasionally have to handle events with team members. After the event stage concludes, using the result and pieces of feedback from the race session, players begin the gameplay loop again, but now with additional information to assist in their decision making.

The Role-Play Character component is used here to model the emotional state and decision making of each team member based on their belief set. The component analyses the actions of the player and determines their effect on the emotional state of each NPC based on their current state and the emotional weighting of the event in their perspective. To give an example,
after each race session, it is possible for a team member to come to the player in order to talk to them. The character might for instance, ask why she was not picked (see Figure 5). Players can then reply back to the team member by selecting from a list of dialogue options. If the player selects an overly aggressive reply, the character is likely to feel angry, affecting its next response.

As mentioned previously, the Role-Play Character component stores the beliefs of every NPC and saves these beliefs over multiple play sessions. These beliefs are related to information such as their last position in the team, skill ratings, opinion ratings and event states. Furthermore, the events sent to the characters are saved, meaning a history of events can be preserved. This allows a history of every team selection to be stored. As all of this information is stored regularly, it can be also be reloaded in further play sessions, allowing for the possibility of a persistent game.

Concerning the Integrated Authoring Tool, this component is used to manage the configuration of the scenario, which contains a list of all possible role-play characters that are dynamically created at the beginning of and during each game. The component also contains all of the dialogue options for the player and the NPCs during various parts of the game, such as team member meetings and post-race events.

V. GAME DEVELOPERS FEEDBACK

Game developers from PlayGen were independent in the integration of the FAtiMA toolkit in their game code and were successfully able to use the toolkit to support the intended gameplay in the two games. They relied on the documentation and examples created for the community and had full access to the toolkit source code. We conducted an informal interview to get their impression regarding the technical integration and the usefulness of the toolkit. Contacts were made by email and face to face. The conversation was around three main questions: (1) How was the FAtiMA toolkit used in the development of the game?, (2) What were the main benefits of using the FAtiMA toolkit? and (3) What were the main difficulties of using the FAtiMA toolkit?

Game developers reported that “the integration was not difficult, but that a proper use of the toolkit requires a steep initial learning curve”. The toolkit facilitated the creation of mechanisms “to determine the change in emotional state and mood depending on the dialogue chosen by the player” and was also useful “to calculate the NPC response to the provided piece of player dialogue, depending on their emotional state and the type of player dialogue selected.” and to “decide how a NPC should greet the player depending on their current relationship with the player.”. They highlighted two main benefits regarding the pedagogical value that the FAtiMA toolkit provided. First, the use of the toolkit was “good because players get immediate implicit (contextual) feedback”. They mean that the emotional responses of the characters were potentially very good cues for the players to assess if they were playing well without the need to show explicit numeric score. The second benefit, was the “ability to dictate the course of conversation indirectly through using the toolkit’s dialogue and NPC emotions systems, as these have made setting up and controlling scenarios a much easier process as a result.”. What is relevant, in the pedagogical sense, is the fact that the definition and setting up of the scenarios was made directly by the trainers who will apply the games. Hence, the game can be configured and adapted by the people who have the most knowledge about the content to be delivered in order to achieve the learning goals of the game.

VI. STUDENTS GAME AI PROJECTS

The toolkit was also put to test in a course on Game AI at IST, University of Lisbon in the fall semester. It was used in the final project of the course (out of 4) that constituted 30% of the grade. Sixty-eight students, working in groups of three, were engaged. They had a workshop on the FAtiMA Toolkit (of about 2 hours) before tackling the problem. They used a version of the toolkit that is integrated with the Unity game engine and uses components to realise the body and expression of the characters developed by other members of the RAGE project.

Each group was given the task of using the FAtiMA toolkit to create two conversational scenarios, one with a single character interacting with the player and another with two characters engaging in conversation with the player at the same time. Students were free to select any theme for the conversation as long as the non-player characters had believable emotional responses and could be configured to have different personalities. All groups managed to finished the project. Some of the scenarios created had quite interesting and surprising themes. For instance, one group chose to create a scenario where players were at the gates of heaven and had to convince the gatekeeper to let them in. To be successful, players had to avoid upsetting the gatekeeper too much. Other groups opted for a more serious theme such as a job interview (see Figure 6) or a shopping scene with a father, his son, and a shopkeeper. With the student’s permission, these scenarios will be publicly available as examples that are part of the toolkit.

From a software quality perspective, given the wide range of scenarios explored by the students, we were able to identify some issues with the toolkit, which were promptly fixed.
VII. Conclusion

In this paper, we argued that the development of serious games is faced with additional challenges that are related to the pedagogical goals that the designers have in mind. For instance, in games that are about teaching conversational skills, developers have to figure out how to offer a rich interaction space that supports the exploration and failure of different communicative actions and their associated socio-emotional effects.

In the mainstream gaming industry, dialogues are typically handled through branching structures that limit the set of possible interactions, by offering little flexibility in the way characters respond to what the players say to them. Alternatively, in the research field of virtual agents, researchers have developed and proposed tools for the creation of conversational agents that have rich socio-emotional models driving their behavior. These agents have great potential for being applied in serious games that teach soft skills, as their behaviors are more procedural and less scripted. However, so far, agent architectures are still far from being widely used in the serious games industry due to, in large part, accessibility issues. With those issues in mind, we took an existing virtual agent architecture, FÁtima Modular, and adapted it to a new toolkit with the goal of making it more appealing to game developers. For that effect, we adopted a functional library pattern instead of a framework-based approach. Moreover, the functionality was divided in two main components, the Role-Play Character and the Integrated Authoring Tool. The first is responsible for managing the character’s beliefs, memories and emotional state as well as running a decision-making process for each character according to its ascribed role. The second component allows the developer to manage the list of all the characters that are available in each game scenario as well as the available dialogues that the characters, including the player’s avatar, can select from at any given state of the interaction.

The resulting toolkit was then applied successfully by a game studio, PlayGen, in the development of two serious games. The first game was designed to teach players how to properly communicate with emotional customers in a customer service setting. The second game has the player managing a sport sails team composed by multiple characters with different role preferences. Both of these games benefited from the use of the toolkit in adding emotional dynamics to their characters that is reflected in their decisions. Additionally, a group of 68 students successfully developed projects for a Game AI course using the toolkit. This experience was a good stress test on the toolkit given the wide variety of scenarios explored by the students.

As future work, we plan to conduct more formal user study centered around the authoring capabilities of the toolkit. The main idea will be to have participants watch a video tutorial about how the toolkit works and then be instructed to change an existing game scenario according to a set of predefined goals. The feedback obtained will then be used to further improve the toolkit.

Acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 and has been partially funded by the EC H2020 project RAGE (Realising an Applied Gaming Eco-System) Grant agreement No 644187.

References
