Accessibility and serious games: What about Entity-Component-System software architecture?

Mathieu Muratet, Délia Garbarini

Accessibility and serious games: What about Entity-Component-System software architecture?

Mathieu Muratet[1] and Délia Garbarini[2]

[1] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France, Université Paris Lumière, INS HEA, 92100 Suresnes, France
mathieu.muratet@lip6.fr
[2] Université Paris Lumière, Paris 8, F-93526 Saint-Denis, France

Abstract. Video games are an integral part of popular culture. The video game industry faces challenges with the increase in players' numbers and application areas including serious games. The increase in the number of players includes disabled players. Therefore, serious games have to consider these audiences who may be affected by one or more temporary or ongoing disabilities. Universally accessible games (UA-Games) aim to create interfaces that can be accessed and manipulated by the largest number of players. Currently few serious games include accessibility features, while accessibility should be considered at the beginning of the serious game design. Then how can we help designers and developers to include accessible features in an existing serious game? To this end, game engines must be efficient but also scalable and modular. This paper studies the interest of the Entity-Component-System (ECS) software architecture to integrate accessibility features in an existing serious game. As a case study, we will take a serious game developed with ECS yet not accessible: E-LearningScape. We will present the accessible features that we have integrated into the serious game and discuss the pros and cons of this approach. This feedback shows us that ECS provides very useful design flexibility to integrate unanticipated interaction features into serious games.

Keywords: Accessibility feature, data oriented programming, Entity-Component-System, serious games, software architecture, universal design.

1 Introduction

Nowadays, video games have become a leisure activity for a large number of people. The Entertainment Software Association announces that 65% of American adults play video games [1]. 54% of them are male and 47% are female. On a global scale, video games represent an object of sharing, socialization, and learning that allows players to interact with the world, to be part of the community and meet people.

However, a large number of people face difficulties in playing video games due to one or more disabilities. Video games stimulate eyesight, hearing and touch, and require players to analyze this multimodal information and perform motor actions in return (i.e.: ability to hold a joystick or perform large movements) [2, 3, 4].

The increase in the number of players includes disabled players whose temporary or chronic impairments reveal inaccessible gaming situations or interaction modalities. To offset this problem, video games have to be adapted to their entire audience. Various approaches attempt to tackle this problem and one is to design universally accessible video games (UA-Games) [2, 5, 6, 7]. This method aims to create many interfaces for everyone, so as to be accessible to as many players as possible with their abilities and disabilities. However, in their survey, Yuan *et al.* [4] found that small number of games incorporated the "design for all" paradigm and that few games implemented a multimodal approach where multiple interfaces are designed for different impairments. One challenge is thus to help designers and developers to update existing games in order to incorporate accessibility features when they have not been anticipated.

To reach this objective, the game engine architecture must be flexible enough, during the game development process, to allow to add new features, to test different interaction modalities and to propose several representations of information. In this paper, we focus on Entity-Component-System (ECS) software architecture. This growing software architecture seems to embed mechanisms favoring modularity and scalability. It is a data-oriented approach based on the notion of composition. Several studies show that ECS can address objected-oriented programming limitations in the field of video games development [5, 8, 9].

Thus, our research question in this paper is: Does ECS architecture help to add unanticipated accessibility functionalities inside a serious game?

In the next section, we will discuss the challenges of adaptation in serious games. Then, in section three, we will present, the ECS software architecture used to develop the serious game studied in this article. In section four we will introduce the methodology we follow in this research and in section five, we will present results about the pros and cons of ECS architecture for integrating new accessibility features, before concluding.

2 Serious games adaptation issues

Serious games are multimodal applications. They require players to have sensory, mental and motor skills. These skills are required to interpret stimuli produced by the game and to control the game through input device control (handle a controller, keyboard, mouse, touchscreen interface, etc.) [2, 3, 4, 10]. Yuan *et al.* [4] identified two categories of stimuli. The first one is essential to achieve a good understanding of the game, as a player who does not perceive these stimuli will not be able to play it. The second one complete the first and is not essential to understanding the game. In the majority of games, primary stimuli are essentially visual while secondary stimuli exploit hearing and haptic (controller vibration) features.

Thus, Bierre *et al.* [6] identified that the inaccessibility of video games is mainly due to the lack of perception of primary and secondary stimuli (because of visual or hearing impairments for instance) but also to the player's ability to analyze these stimuli and provide new actions in return. An overload of visual and hearing stimuli can place players with cognitive impairment in serious difficulty processing information and deciding

which answer to provide. A motor impairment will limit the player's abilities to carry out the response if it results in in-game control manipulations, especially in a limited time context.

Much research contributes to producing knowledge on video game accessibility. Some studies analyze the difficulties met by players [4] and others focus on new functionalities to make video games more accessible [2]. The improvement of video game accessibility is also inspired by the standards initially developed for websites (WCAG[1] et W3C[2]). There are indeed similarities, in particular for interfaces and menu management (button spacing, alternative texts, subtitles, animations, contrast modification, text-to-speech). More specifically, in the field of video games, the Game Accessibility Guideline [11] initiative offers a full list of recommendations to improve the accessibility of video games according to the type of impairment.

The UA-Games approach and the works mentioned above are in line with research conducted in TEL (Technology Enhanced Learning) and especially on the three dimensions of adaptation [12] (adapted, adaptable and adaptive). Applied to the field of serious games, this theory allows us to define adapted, adaptable and adaptive serious games.

Adapted serious games target a particular profile of players. Interactions are set up during the design phase. As the environment creates disabilities, a serious game adapted to a particular impairment may consequently generate inaccessibility for other players. The serious game "A blind Legend" [13] is a singular example of this. This serious game puts the player in the shoes of a blind character for whom the only channel of communication used is sound. This serious game is therefore by nature inaccessible to players with a hearing impairment.

Adaptable serious games are the most widespread. Recent serious games integrate more and more menus allowing players to customize the interaction modalities as they wish (moving speed of the pointing device, colors/contrasts, keys/control buttons, difficulty level, etc.). These settings allow all players (including disabled players) to adapt the serious game to their skills in order to improve their playing experience.

Adaptive serious games have been less common because they have to include models (player's model and/or model of challenge resolution process) in order to automatically adapt interfaces and content to the difficulties encountered by the players.

Ergonomics research on activity theories provide knowledge that can help to understand how players master serious games. Activity theories aim to understand a subject's activity when trying to achieve a goal [14]. So, activity is different from a prescribed task. A task specifies what needs to be done (i.e. the goal) and the procedure to achieve this goal. An activity is singular, located, finalized and mediatized by an instrument [15]. Here, an instrument refers to an artifact associated with schemes. According to activity theories, we consider that a video game engine is an artifact on which the players will build schemes to interact with it. These schemes can only be built if the player is able to access the primary (or even secondary) stimuli and provide actions in return. A serious game uses an engine (the artifact) and offers objectives to the players in the

[1] WCAG = Web Content Accessibility Guidelines
[2] W3C = World Wide Web Consortium

form of a set of missions or quests to complete or a record to beat. The rules of the serious game constrain the player's actions and make it possible to calibrate the complexity of a game situation. Thus, a serious game can be abstracted as a task according to the activity theories. The player's activity is therefore what is accomplished by the player according to his/her skills and it depends on the accessibility of the engine and the complexity of the situations offered (including goals). This incremental process named instrumental genesis allows players to appropriate the artifact (transforming artifact into instrument) and involves two types of transformation: instrumentalization, where the player will adapt the artifact to his/her needs (dimension of adaptable serious games) and instrumentation, in which the artifact influences the player's actions (dimension of adapted and adaptive serious games).

Thus, improving serious game accessibility consists in supporting the instrumental genesis defined in activity theories and working on the three dimensions of adaptation (adapted, adaptable and adaptive).

3 Entity-Component-System: a software architecture coming from video games

Video game and serious game developments are mainly based on the object-oriented programming paradigm (OOP), whose C#, C++ and Java are the most used programming languages. However, OOP principles such as encapsulation, message sending and inheritance can make game engine maintenance and scalability difficult. With an object-oriented approach, developers model game elements in the form of classes that may be specialized into subclasses, etc. The video game development process is highly iterative, adding new game mechanisms or interaction modalities may request changes in the modeling initially designed. Then, these modifications may have significant consequences on developments and require code refactoring, which is expensive in development time and a possible source of bugs. The rigidity of the inheritance tree is a hindrance in this context.

The Entity-Component-System (ECS) is a software architecture mainly used for developing video games [16, 17]. This architecture uses a data-oriented approach and is built around three concepts. Entities (first concept) represent game objects but do not contain either data or methods. An entity is a simple reference to a collection of components (second concept) that contain data. Components describe an entity's aspects such as its color, size, speed, etc. A component may be added or removed dynamically to an entity. Systems (third concept) define the game logic. They access the components of the entities in order to process and update them. They modify the game data and then update the simulation.

ECS is a software architecture in which simulation is data-driven. ECS is based on composition whereas the object-oriented approach focuses on encapsulation and inheritance. ECS was developed to answer two issues: improving computer code modularity and improving game engine performance. As for modularity, the data-driven approach allows to add new game mechanisms or interaction modalities with a limited impact on

the existing code. To integrate a new feature, the developer has to (1) define the components required to store the data, (2) add these components to the entities concerned and (3) implement systems that will process these components. Contrary to classes in OOP that contain data and logic, in ECS they are separated: data in components and logic in systems. From a performance point of view, ECS allows to control the organization of data in memory and optimizes access to components. In this paper, we are studying ECS for its modularity promises and we want to evaluate it to add accessibility features. We are not hereby studying questions of optimization.

Garcia *et al.* [5] also address this architecture based on component and data-driven approach for the development of accessible video games. They study the interest of ECS to create different game object representations that can be changed in real-time and without consequences on the game logic. Authors create components that contain the data of the different stimuli (graphic, audio, haptic) and specific systems to manage these components. When the game is running, the modal presentation sent to the player depends on the components attached to a game entity. It is therefore possible to switch from one representation to another in real-time, according to the needs and abilities of the player, offering a more accessible game experience. Moreover, this research highlights the advantage of this architecture for using an external data source (XML file) in order to define game-specific settings. Players could thus access this file and create a profile by customizing entities, modifying presets and creating default configurations adapted to their needs.

The runtime ECS advantages promoted by Garcia *et al.* [5] confirms our hypothesis that ECS architecture is adapted and useful to more easily modify an existing video game whose accessibility was not initially considered during the design step.

4 Methodology

As a preamble to this section let us introduce the artifact we have been working on. E-LearningScape (see Fig. 1) is a numerical adaptation of a physical escape game [19]. In this numerical adaptation, the participants (in teams of 2 to 5 players) play the role of a sandman immersed in the dream of Camille, a young lecturer on the eve of teaching her first lesson. Their challenge will be to help Camille structure her thought in her dream by solving puzzles using pedagogical concepts. The team members gather around a computer. One player controls the game and moves inside the virtual universe in a first-person navigation to discover fragments of dreams giving access to material in the real world. All members of the team solve puzzles inside and outside the game, these two facets feeding each other. E-LearningScape has several objectives, the main one being to test knowledge in the field of education and the other ones to promote team work and cohesion.

E-LearningScape was initially developed without accessibility constraints in mind. It is an open-source and totally designed with ECS architecture (it was developed with Unity and the FYFY plugin [18]).

Fig. 1. E-LearningScape screenshot

This serious game is an interesting case study for our research question, which is: Does ECS architecture help to add unanticipated accessibility functionalities inside a serious game?

To tackle our research question, the serious game was analyzed with the Game Accessibility Guideline grid [11]. These guidelines are divided into six categories (motor, cognitive, vision, hearing, speech and general), each one broken down into three levels (basic, intermediate and advanced). This analysis, performed and discussed by two experts in accessibility, aimed to recommend accessibility functionalities suitable to the serious game. Based on these recommendations, a game developer incorporated the new functionalities inside the serious game. This developer was not part of the original team that developed the serious game and was not familiar with ECS architecture. The developer got support from an expert developer who was a member of the initial project. Each new functionality was studied to evaluate the advantages and drawbacks of ECS architecture. For each of them, the developer reflected on the following questions: Is it possible to integrate the functionality only using Unity editor (without a line of code)? How many Unity editor manipulations are required with a classical Unity script to integrate the functionality? Does a system in which the functionality can be integrated exist? Which components and systems have to be created to integrate the functionality? Which choices make future updates easier?

5 Results

5.1 E-LearningScape analysis

Among the 121 recommendations included into the Game Accessibility Guideline grid [11], experts in accessibility have retained 67 criteria that make sense for E-LearningScape. For instance, they excluded recommendations on voice subtitles, multiplayer and chat features, and virtual reality and mobile devices because they were judged not consistent for the serious game. Among the 67 criteria identified by experts in accessibility, 10 were already integrated into the game (for instance: including tutorials, allowing reminders of current objectives during gameplay, ensuring no essential information is covered by sounds alone, etc.). The developer updated the game and added 34 supplementary recommendations to fulfill the basic level of motor, cognitive and

vision categories (hearing and speech categories are not requested by E-Learning-Scape). Let us give some examples of new features that were incorporated: supporting more than one input devices, including options to adjust the sensitivity of controls, using simple clear text formatting, ensuring no essential information is covered by color alone, etc. The current version of the game includes 17 basic recommendations over the 18 suitable ones, 22 intermediate recommendations over the 35 suitable ones, and 5 advanced recommendations over the 14 suitable ones. Fig. 2 shows details about accessibility completion of the game before and after the updates.

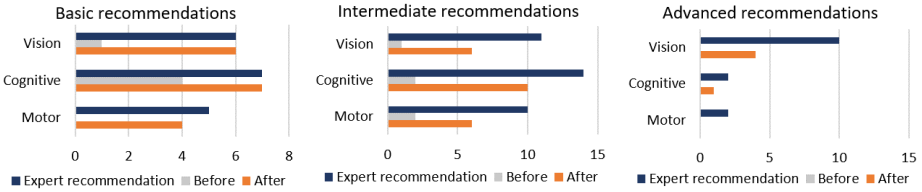

Fig. 2. Accessibility recommendations integrated into the game

5.2 Adapted serious game

Modifications have been made to change serious game interfaces and interactions. To complete the default mouse/keyboard inputs, the developer increased the diversity of control devices: the serious game is now playable with the Xbox one controller, the Microsoft Xbox adaptive (XAC), the Nintendo Nunchuck and specific controllers. The Unity environment provides native tools to abstract controllers. In this case, ECS did not bring any advantages. As for controllers' abstraction, the developer reviewed menu navigation and added visual effects to help players to easily identify the UI element focused on. Similarly, Unity's native tools are very efficient to define interface navigation order and highlight interactive UI elements. Other functionalities have been integrated without the ECS layer because they were relating to static graphical parameters: color review (limitation of red and green colors), black outline on textual elements, etc. All these accessibility features were editable inside Unity's editor and did not require programming.

In the first version of the serious game, in-game interactive objects were highlighted with visual effects. Experts' recommendations suggested dubbing this information with audio notifications. With a classic use of Unity, the developer would create a new script, add it manually to all interactive game objects and configure them in Unity inspector with the risk of forgetting one of them (the game contains more than 60 interactive game objects). In this case, ECS enables to easily add this functionality. Indeed, a system already existed to highlight interactive game objects. The developer just added one line of code in this system to dynamically add a new component on the highlighted game objects. This new component contains sound data to play. Then he created a new system to process sound components and play audio notifications. The developer could play sound directly inside the existing highlighter system, yet chose to separate it inside

a new system. This makes the audio notification system independent from the highlighter system. Thus, the audio system can be turned off by the player without any consequence on visual notifications. Moreover, the audio system may be used in other cases that require audio notifications (changing room, enigma success/fail, etc.). The consequences of adding this new functionality to the existing code were very limited.

ECS was also useful to easily identify the different functionalities of the game that a developer could change. Indeed, each functionality is implemented inside a system, and updating a functionality consists in identifying the concerned system and focusing attention on it. Thus, the developer, who did not master ECS at the beginning of this work found it easier to study the list of systems to understand the game logic than parsing game objects hierarchy and components interdependences. He updated the system that manages interactive game objects to enable the player to hold down or successively press a button; or to drag and drop a game object without holding down a button. Finally, he modified the system that manages the camera to add a zoom function and to switch point of view (first/third person).

5.3 Adaptable serious game

Several recommendations concern menus, to enable players to personalize serious game interfaces according to their needs. The developer added functions to change controls, graphics, and sounds. For instance, the player can disable/enable animations, change moving speed and camera rotation speed, adjust music and effect volumes, grow cursor, and update scene luminosity. For all of these new features, the default tools of Unity were sufficient.

On the other hand, ECS was useful to apply massive changes to game objects. One recommendation is that players have to be able to change text font, to switch from the default non-accessible font to an accessible one. With a classical usage of Unity, the developer would have added a new script to each textual game object or created a text manager and bound each textual game object in (more than 570 game objects are concerned). In these two solutions, designers risk making errors when a new text is added to the game, and forgetting to add the accessible script to the game object or to bind the game object to the text manager. With the ECS approach, the developer added a new system and created a family to filter game objects that contain default text components. When the player switches text font, the system updates all textual game objects in the scene. Adding new accessible text into the game requests designers to simply add a default textual game object to the scene without taking into account any accessibility constraints (no script to add, no specific manager to bind). The same principle has been applied to manage windows' transparency.

5.4 Adaptive serious game

The adaptive dimension is present in this serious game with a monitoring module. This functionality generates in-game hints to help players depending on their difficulties. The monitoring module integrated into the game is compatible with ECS architecture

[20] and enables to monitor players' activities either at the level of entities (local monitoring) or at the level of families (set of entities – global monitoring).

Integrating this monitoring module requires a set of modifications in the existing code, especially to produce tracks. The first challenge was to identify inside existing programs in which the player's actions were validated. ECS architecture helps to understand game design because each logic functionality is implemented inside a specific system. Then the developer easily identified systems that manage player inventory, moving, interactive objects, etc. Finally, producing tracks consisted in updating systems and adding one line of code to build the monitoring component (provided by the monitoring module) with action data. This monitoring component is processed by the monitoring module that labels the player's actions (correct, too late, useless, etc.).

The second step was to create a new system to process results from the monitoring module. This system chooses the next hint to be displayed to the player depending on labeled actions aggregated over time, player progression, and time left. Generated hints may suggest searching for a specific area, indicate a clue position, explain a bad answer, and validate the fact that players have all the elements to resolve an enigma.

6 Conclusion

We began this paper by pointing out the importance of taking into consideration the question of accessibility to serious games. We were interested in the cases in which accessibility had not been taken into account in the early phases of development and had to be integrated retrospectively. So, we studied the potential benefits of ECS architecture in this context. Our theoretical foundation is based on activity theories and the three dimensions of adaptation (adapted, adaptable and adaptive).

We analyzed the serious game E-LearningScape and we added 34 supplementary recommendations from the Game Accessibility Guideline grid [11].

The contribution of this paper concerns the analysis of updates made into the serious game E-LearningScape. We found that Entity-Component-System architecture does not bring added value to process singular and/or static game objects. In this case, the classical features of Unity seem to be more effective. However, this work showed that ECS is advantageous to carry out transformations impacting a large number of game objects and to avoid repetitive manipulations that are a source of errors. We also found that the principle of modularity and decomposition of the game's features into systems, allowed the developer to quickly identify the parts of the program to be modified.
Thus, the Entity-Component-System architecture seems to be interesting to help integrating accessibility features into serious games, especially when these have not been anticipated. In this research we worked on a serious game initially designed with ECS. However, we think that this approach is also practical for non ECS serious games made with Unity because FYFY and Unity perfectly fit together and complement each other. We plan to check this hypothesis in future research.

Acknowledgements. We thank Séverine Maillet for comments on the paper.

References

1. ESA Essential Facts About Computer And Video Game Industry, https://www.theesa.com/esa-research/2019-essential-facts-about-the-computer-and-video-game-industry/, last accessed 2020/03/19.
2. Grammenos, D., Savidis, A., Stephanidis, C.: Designing Universally Accessible Games. ACM Computers in Entertainment, vol. 7, no. 1, pp. 29 (2009).
3. McCrindle, R. J., Symons, D.: Audio space invaders. In: Third International Conference on Disability, Virtual Reality and Associated Technologies (2000).
4. Yuan, B., Folmer E., Harris, F.: Game accessibility: A survey. Universal Access in the Information Society, vol. 10, pp. 81–100 (2011).
5. Garcia, F. E., de Almeida Neris, V. P.: A Data-Driven Entity-Component Approach to Develop Universally Accessible Games. Universal Access in Human-Computer Interaction, vol. 8514, pp. 537–548 (2014).
6. Bierre, K., Chetwynd, J., Ellis, B., Hinn, D. M., Ludi, S., Westin, T.: Game Not Over: Accessibility Issues in Video Games. In: Human-computer interaction (2005).
7. Grammenos, D., Savidis, A., Stephanidis, C.: Unified Design of Universally Accessible Games. In: Universal Access in Human-Computer Interaction, Beijing (2007).
8. Rafaillac, T., Huot, S.: Polyphony: Programming Interfaces and Interactions with the Entity-Component-System Model. In: 11th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Valencia, Spain (2019).
9. Gestwicki, P.: The entity system architecture and its application in an undergraduate game development studio. In: International Conference in the Foundations of Digital Games, (2012).
10. Bierre, K., Hinn, M., Martin, T., McIntosh, M., Snider, T., Stone K., Westin, T.: Accessibility in Games: Motivations and Approaches. In: The International Game Developers Association (2004).
11. Game Accessible Guidelines, http://gameaccessibilityguidelines.com/full-list/, last accessed 2020/03/11.
12. Hussaan, A. M.: Generation of Adaptive Pedagogical Scenarios in Serious Games. Université Lyon 2, Lyon, France (2012).
13. A Blind Legend, http://www.ablindlegend.com/, last accessed 2020/03/11.
14. Daniellou, F., Rabardel, P.: Activity-oriented approaches to ergonomics: some traditions and communities. Theoretical Issues in Ergonomics Science, vol. 6, no. 5, pp. 353–357 (2005).
15. Forcisi, L. A., Decortis, F.: Children's Creativity at School: Learning to Produce Multimedia Stories. In: Congress of the International Ergonomics Association (2018).
16. Bilas, S.: A Data-Driven GameObject System. In: Game Developers Conference (2002).
17. Capdevila, B.: Serious game architecture and design: modular component-based data-driven entity system framework to support systemic modeling and design in agile serious game developments. Université Pierre et Marie Curie, Paris, France (2013).
18. FYFY, https://github.com/Mocahteam/FYFY, last accessed 2020/03/11.
19. LearningScape, https://sapiens-uspc.com/projets-innovants/learningscape-2/, last accessed 2020/03/11.
20. Muratet, M., Yessad, A., Carron, T.: Understanding Learners' Behaviors in Serious Games. In: Advances in Web-Based Learning – ICWL 2016, Rome, Italy (2016).