EDICULA

Educational Digital Innovative Cultural heritage related Learning Activities

Project Code: 2020-1-EL01-KA203-079108

NATIONAL TECHNICAL UNIVERSITY OF ATHENS [GREECE]

SAPIENZA UNIVERSITA DI ROMA [ITALY]

BEZALEL ACADEMY OF ARTS AND DESIGN [ISRAEL]

PerpetielSI SRL

PERPETIELSI SRL [ROMANIA]

ISRAEL ANTIQUITIES AUTHORITY [ISRAEL]

HELLENIC RESEARCH INSTITUTE OF THE ALEXANDRIAN CIVILIZATION [GREECE]

INTELECTUAL OUTPUT:	**O4 EDICULA DIGITAL GAMES**
DELIVERABLE:	**D4.2 EDICULA System Architecture**
LEAD ORGANIZATION:	PerpetielSI
DATE:	31 January 2022

Co-funded by the Erasmus+ Programme of the European Union

Table of contents:

1. Introduction

The term Serious Game includes a wide, heterogeneous field of digital games with varying purposes and objectives and for a multitude of different application areas (Söbke et al., 2016). This deliverable gives an overview on the technical aspects of serious games including certain architectures and engines that are available for the development of digital games. Moreover the basic principles and requirements for serious game software are covered, while selected software architectures and examples for game engines are presented in addition with selected components in order to make the appropriate choices for the EDICULA Digital Game.

2. Serious Game Engines and Architectures

Generally, serious games are a subset of digital games, which are a significant part and driving factor of the creative industries nowadays. It is common practice to describe game development by comparison to software development. According to Murphy-Hill et al. (2014) in game development the requirements to the product are more unclear compared to conventional software development and therefore, requirements often are subject to change during the development process. This finding is backed by the complex, non-deterministic and non-linear but iterative process of game design in order to develop working, fun-creating game mechanics. Cooper and Scacchi even underline that game development is a broad and comprising field, which leads to developers being narrowly skilled in probably only one game genre.
Furthermore, serious game development has the additional burden of integrating the “serious” element into the game, making the process even more challenging as it removes degrees of freedom from the design process in order to achieve a seamless integration of content and game. Among the differences to conventional game development is that the hardware requirements need to be rather low, since educational applications are often connected to users that do not always have the most recent hardware available. Moreover, target systems, users and target groups of serious games are more heterogeneous and not necessarily gamers or familiar with technology. This usually leads the developers to more simple and user friendly equipment and user interface. Another characteristic of serious games is the necessity of more accurate, realistic and appealing simulation models and virtual environments.
In conclusion, serious game development is a complex task in a technically fast moving environment. Technological progress also boosts the capabilities of commercial entertainment games that can be considered as benchmark for attractiveness of serious games.

2.1 Architectures

Serious games as complex software applications often represent configurations of multiple software components, libraries, and network services. Therefore serious games must have an architecture and there are at least four kinds of computer games software architecture in general that arise mostly in networked multiplayer games: (a) the static and dynamic runtime architectures for a game engine, (b) the architecture of the game development frameworks or SDKs that embed a game’s development architecture together with its game engine, (c) the architectural distribution of software functionality and data processing services for networked multiplayer games, and (d) the informational and geographical architecture of the game levels as designed play spaces. In contrast, the focus on CG as interactive media often sees little/no software architecture as being relevant to game design, especially for games that assume a single server architecture or PC game runtime environment, or in a distributed environment that networking system specialists are assumed will design and provide.
Kruchten (1995) forms a 4+1 view model of architecture stating that software architecture deals with abstraction, composition and decomposition, as well as style and aesthetics (Figure 1).

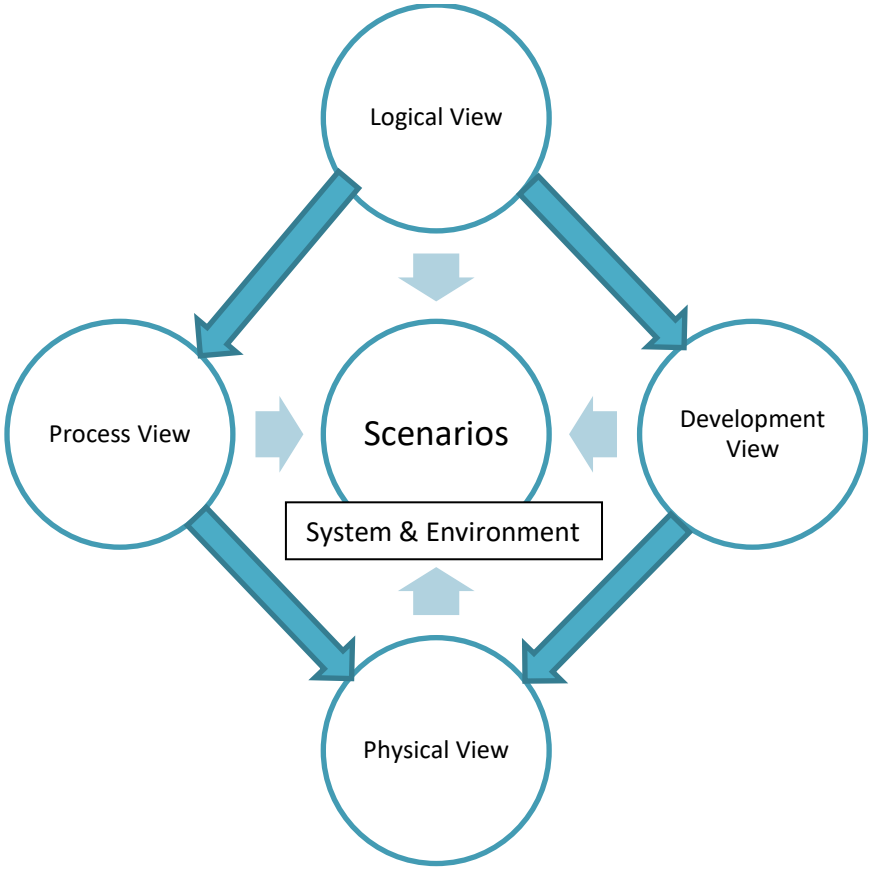

Figure 1: 4+1 view model of architecture by Kruchten

According to Kruchten there are four views of a software architecture to describe the above model: (i) a logical view that illustrates the object model, (ii) a process view that deals with concurrency and synchronization issues, (iii) a physical view that describes the "mapping of the software onto the hardware and reflects its distributed aspects" and (iv) a development view which represents the software's static organization in its development environment. Bass et al. (2003) evolve Kruchten's view model by adding the concepts of components, their interfaces and their interrelated compositions. As far as game engines are concerned, the concept of software architecture and its components becomes quite obvious as shown in Figure 2. It is suggested that the logical view (i) and the process view (ii) of Kruchten's model are handed by the game engine's architecture, meaning that object models have to follow the game engines' technical conventions and within their game loop they frame the handling of synchronization and concurrency issues. The physical view (iii) and the development view (iv) are applicable to the game architecture itself, while the distribution of the various components of the hardware is handled by the physical model. The development model on the other hand describes the integrated components and libraries, differentiating the architecture of a game engine and the architecture of a game itself.
Apart from game engines, there are further components that are included in a serious game architecture, such as platforms, sensors, servers, authoring tools, content management systems, client-server-middleware, etc. All the above components should be taken into consideration for the architecture design and the development of a serious game.

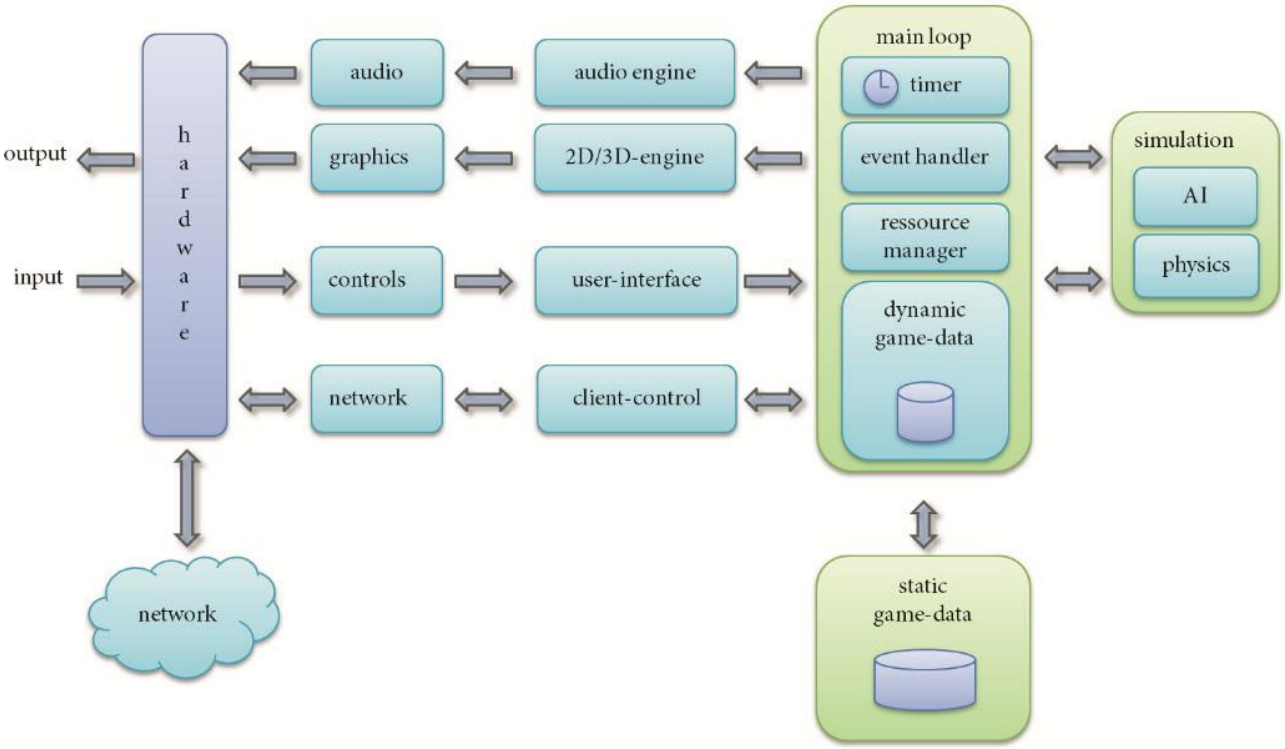

Figure 2: Game engine: Software architecture as a composition of components and interfaces (Masuch et al., 2015).

The EDICULA Digital Games aims to exploit all the semantic data and available content to create a realistic and immersive 3D virtual environment for "edutaintment" purposes by developing a serious game consisting of multiple mini games such as virtual tours, digital class and quiz games. The design of the static aspect of a serious game like that is comprised of two tightly coupled tasks: the architectural design of the virtual environment (virtual tour & digital class) and the design of the available content, information and presentation of all the displayed objects/ settings.
The environmental design aims at supporting the user in navigating, while maintaining a sense of orientation within the environment even when the user is not familiar with technology, especially VR and AR. In order to achieve this, architectural knowledge has proved invaluable during the design and development of all spatial elements, for enhancing users' environmental knowledge and for directing participant attention towards certain points of interest. The virtual environment itself can be characterised as a large, dense and relatively static area, that offers at the same time multiple interaction opportunities for the user. In virtual tours, the user's navigation is primarily explorative, although it is important to make the experience more appealing and interactive by implementing a technique that allows him/her to manipulate the available objects/ info points. These are the main guidelines for the development of the EDICULA Digital Game and its architecture.

2.2 Game Engines

A game engine is a software framework primarily designed for the development of video games, and generally includes relevant libraries and support programs. Game engines are middleware that allows the integration of multiple resources turning them into assets with graphic information, programmed content and physical simulation parameters. These features meet the requirements for modelling, visualizing, simulating and controlling complex production systems (Zarco et al., 2021). The basic functionalities of game engines include the following attributes: graphics rendering, simulation of mechanical behavior, kinematics, momentum, and collision detection generally through a physics engine, scripting integration, multithreading capability, scene management, integration of resources (animations, sounds, images, textures, etc.),

communication capability through different protocols, artificial intelligence, networking and multiplayer, as well as the integration of assets, plugins and cross compiling (Figure 3).

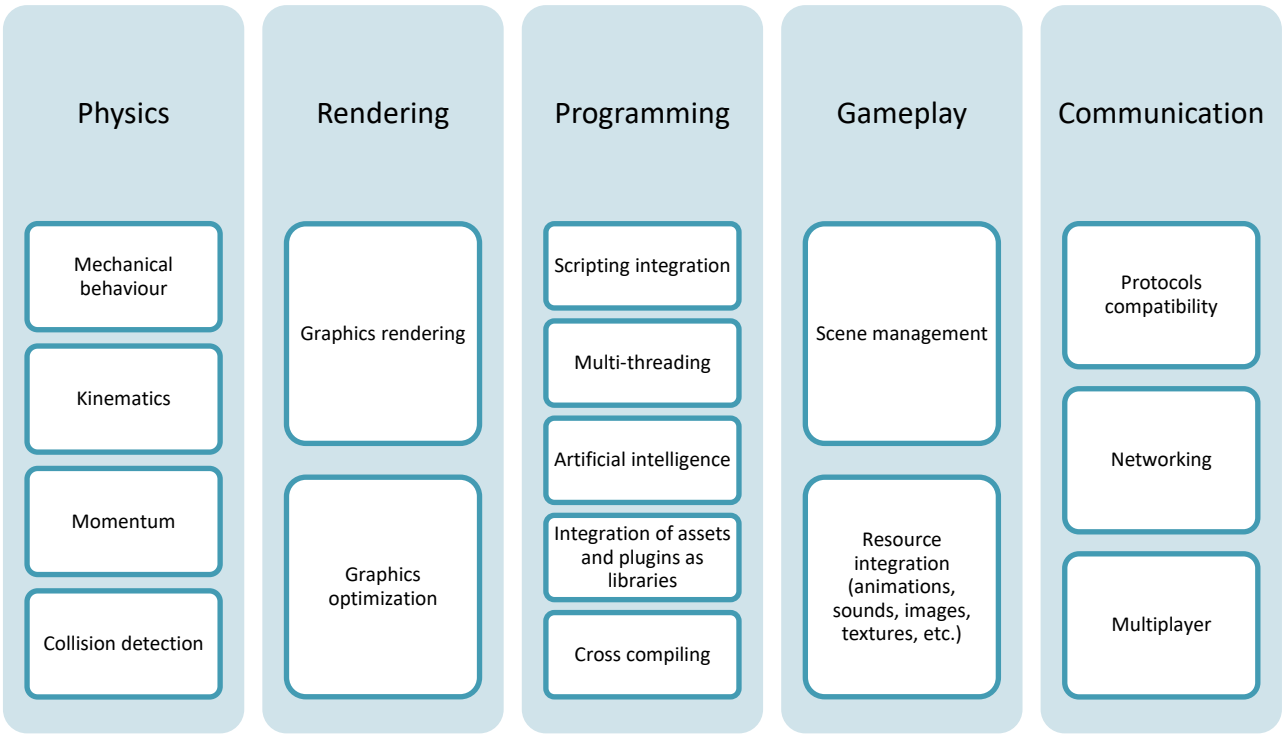

Figure 3: Main attributes of game engines.

There is no formal classification for the available game engines. However there are the following segments for game engines:

- By the development environment (2D, 2.5D, 3D)
- By the app base (PC, Mobile, Console, Internet, TV, others)
- By business model (open source, freeware, fix priced license, based on income)

In literature, there are several comparisons about game engine performance (Vasudevamurt et al., 2015 & Zarco et al., 2021 & Xie J., 2019 & Uskov et al., 2014) focusing on the basic features of each engine, the price, support, flexibility, interoperability, usability, system requirements, usability, functionality, multimedia support, working environment, etc. More than 70 serious games were analyzed (Vasudevamurt et al., 2015) according to the game engine that was used for their development and the area of interest (education, simulation, virtual reality) and are presented at the table below showing the number of serious games developed with each engine.

Game Engine	Educational Serious Game identified	Simulation Serious Game identified	Virtual Reality Serious Game identified
Unity	12	10	4
Torque	13	2	5
Unreal Engine	1	4	5
Unigine	0	6	4
Neoaxis	3	2	0
CryEngine	0	2	0

Table 1: Serious Game Engines used for Serious Game development in selected areas.

Game engines enable the implementation and use of design elements and techniques, game mechanics and analytics and software architecture in serious applications to improve user experience, engagement, effectiveness and productivity. Furthermore, the use of game engines for industrial and engineering applications is increasing, boosting the development of them especially for their technical characteristics and the improvement in physical simulations, as well as the optimization of rendering and latency times (Zarco et al., 2021). Gamification for controlling ultra-flexible production systems through game engines leverages elements of video games and their development environments in the technical context of the factory with the goal of creating precise control and a positive experience for the user. By creating a heightened visual experience, operators can effectively perform tasks and even build new skills. Game engines can exploit resource redundancy, enabling efficient integration of new and diverse production resources into simulations. These resources can be scaled or detailed according to the needs of the simulation. In addition, the integration of metadata and reliable physical simulations enables the generation and evaluation of valuable engineering parameters. Game engine compatibility enables the development of applications for numerous end-user devices, increasing the agility of information in the static and dynamic systems of a factory. In this way, employees can have the information they need at the right time and in the right place. At the table below Zarco et al. (2021) present a summary of meta-analysis and self-research about game engines features and performances giving a thorough insight on game engines and potentials.

Game Engine Features	2D/ 3D	Work flow editor	World editor	Physics Engine	AI support	Networ-king online	Progra-mming skills	Scripting	Platform	License
Unity3D	2D/ 3D	Yes	Yes	PhysX, Box2D, Unity Physics and Havok	i.a. Bots and FSM	Yes	Low	C#, JavaScript	PC, Web, Mobile, Xbox, PS	Free, Paid
Unreal Engine	3D	Yes	Yes	Chaos Physics	i.a. FSM	Yes	Low	C++, Blue-prints	PC, Mobile, Xbox, PS	Free, Paid
Cry-Engine	3D	Yes	Yes	Proprietary	Yes	Limited	Low	C#, C++, Lua	PC, Mobile, Xbox, PS	Free, Paid
Game Maker	2D	Yes	Yes	Box2D and Liquid-Fun	Yes	Yes	Low	GLM	PC, Mobile, Xbox, PS	Free, Paid
Neoaxis Engine	3D	No	Yes	Bullet	Yes	Limited	Medium	C#	PC	Free
Game-Salad	2D	Yes	Yes	Rigid-body physics	Yes	Yes	Low	Lua	PC, Mobile, Web	Free, Paid
Cocos2D	2D	No	Yes	Chip-munk	Yes	Limited	Medium	C++, JS	PC, Mobile	Free
Torque	2D/ 3D	No	Yes	PhysX	Yes	Yes	Medium	C++	PC, Web	Paid
Unigine	3D	No	Yes	Proprietary	Yes	Limited	Medium	C#, C++	PC, Web, Mobile, Xbox, PS	Paid
Quake 4	3D	Limited	Yes	Proprietary	Yes	Limited	High	C++	PC	Free
Construct2	2D	Yes	Yes	box2dweb, Cocoon-JS	Yes	Yes	Low	JavaScript	PC, Mobile, Web	Paid
Shiva 3D	2D/ 3D	Yes	Yes	ODE physics engine	Yes	Yes	Medium	Lua, C++, ObjectiveC	PC, Xbox, Mobile, PS, Web	Paid
Cafu (MIT)	3D	Limited	Limited	Proprietary and Bullet	N/A	Yes	Medium	Lua	PC, Mobile	MIT Free
Amazon Lumber-yard	3D	Yes	Yes	PhysX	N/A	Yes	Medium	C++	PC, Web, Mobile, Xbox, PS	Free
Panda-3D (Disney)	3D	5	5	Proprietary, Bullet, PhysX	i. a. FSM	Yes	High	C++, Python	PC	BSD but N/A
Delta 3D	3D	Yes	Yes	ODE	Yes	Limited	Medium	C++	PC	Free
Source Engine	3D	Yes	Yes	Havok, Rubikon	Yes	Limited	Medium	C++	PC, Xbox	Paid
Frost-Bite	3D	N/A	N/A	N/A	Yes	Limited	Low	N/A	PC, Web, Xbox, PS	N/A
SnowDrop	2D/ 3D	N/A	N/A		Yes	Yes	Low	N/A		
Dunia 2	3D	No	Yes		Yes	Yes	Low	N/A		
Fox	3D	Limited	Limited		Yes	Yes	Low	N/A		
Chrome Engine	3D	Limited	Limited		Yes	Yes	Low	N/A		Removing Support
ID tech 5/6	3D	No	Limited		Yes	Yes	High	C++		N/A
Adventure Game Studio	2D	N/A	N/A	None	No	Limited	N/a	JAVA, C#	Windows	

Table 2: Summary of meta-analysis and self-research about game engines features and performances.

Further research was conducted by Cowan B. & Kapralos B. in 2014 in terms of game engines and frameworks leading to the results presented below in Figures 4 & 5.

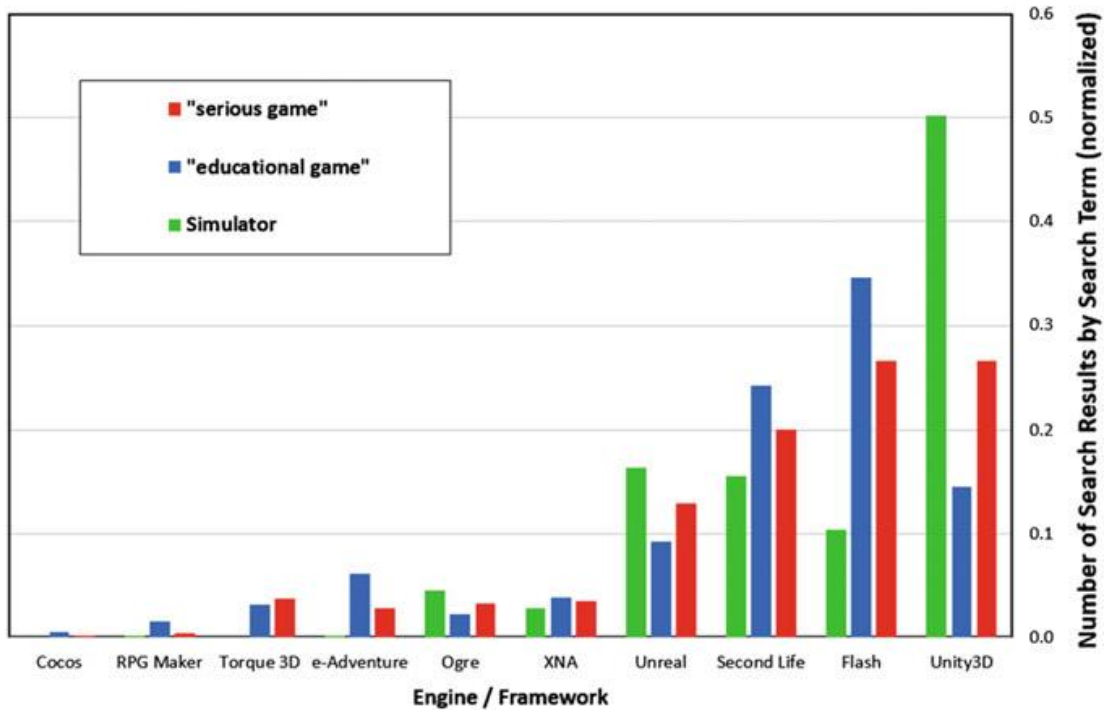

Figure 4: Game engine/framework versus normalized (number of search results) by search term.

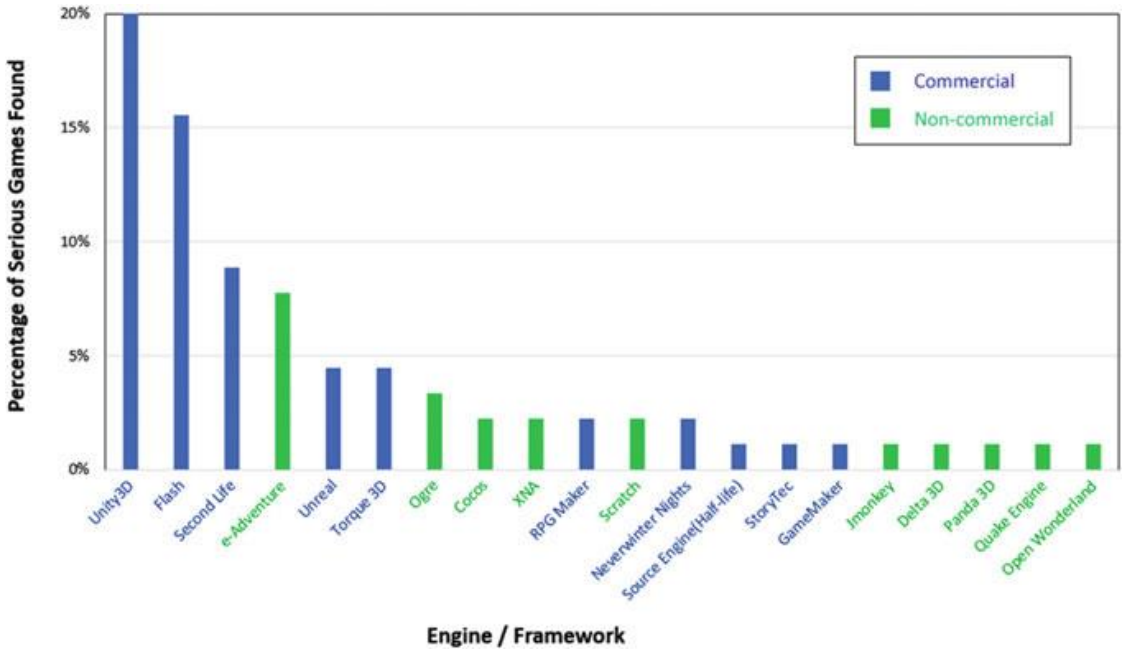

Figure 5: Game engine/framework versus the percentage of serious games that were created with it.

According to literature and the above comparisons it was decided to further explore the options of Unity3D and Unreal Engine, as game engines for the development of the EDICULA Digital Game. First of all, both game engines have a project and version management (Unity with Unity Hub and Unreal with Epic Games). This is quite helpful if a developer needs to use components from previous versions and make the necessary

modifications. In Unity3D there is the option of both 2D and 3D new project creation, while in Unreal there is only the option of 3D project templates. Furthermore, when importing 3D game objects into Unreal the engine separates the hierarchy of components without grouping them into one game object, while Unity3D keeps the hierarchy. Unity3D automatically assigns the metadata corresponding to the materials imported into a project using shaders and materials, while Unreal is not working the same way. In addition, both game engines are rapidly evolving the past years to match even more accurate physics simulations.

GAME ENGINE	UNITY3D	UNREAL
Functionality	Both game engines can integrate functions and hierarchies for the accumulation of production modules, as well as the corresponding programming to simulate their behavior	
Scalability	Both game engines allow spatial scaling of the elements, as well as project scaling	
Physics simulation accurateness	Uses an integration strategy of different physics engines for a realistic and relatively accurate simulation	Concentrates on collisions and their visual effects
Resource redundancies	Both game engines allow resource redundancy	
Highly complex system visualization	GUI visual effects, 2D/3D models (in various formats) and environments	High quality of visual effects for 3D models and environments

Table 3: Summary of scopes and limitations of Unity3D and Unreal (Zarco et al., 2021).

3. EDICULA Digital Game Architecture and Game Engine

Based on the above research it is important to mention that choosing the appropriate game engine is not an easy task when developing a serious game. Unity3D is highly recommended for serious game developers both in the industry and in academia and since there is a previous experience with the development of serious games and digital applications in Unity3D, it was decided to develop the EDICULA Digital Game in Unity3D v.2017.2.0f3. Below in Figure 6 the EDICULA Digital Game Architecture is presented and in Figure 7 the analysis phase for the design of the game.

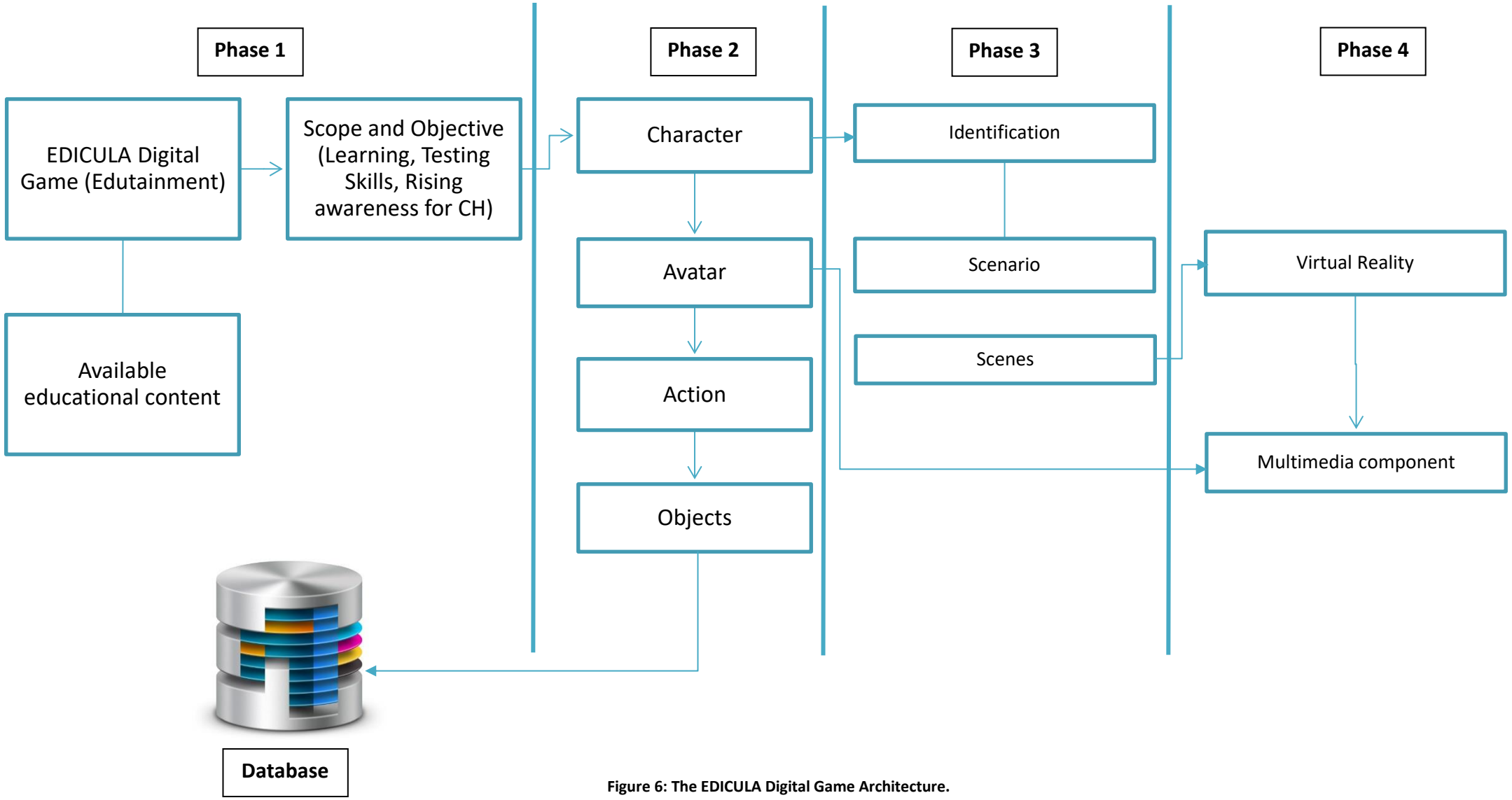

Figure 6: The EDICULA Digital Game Architecture.

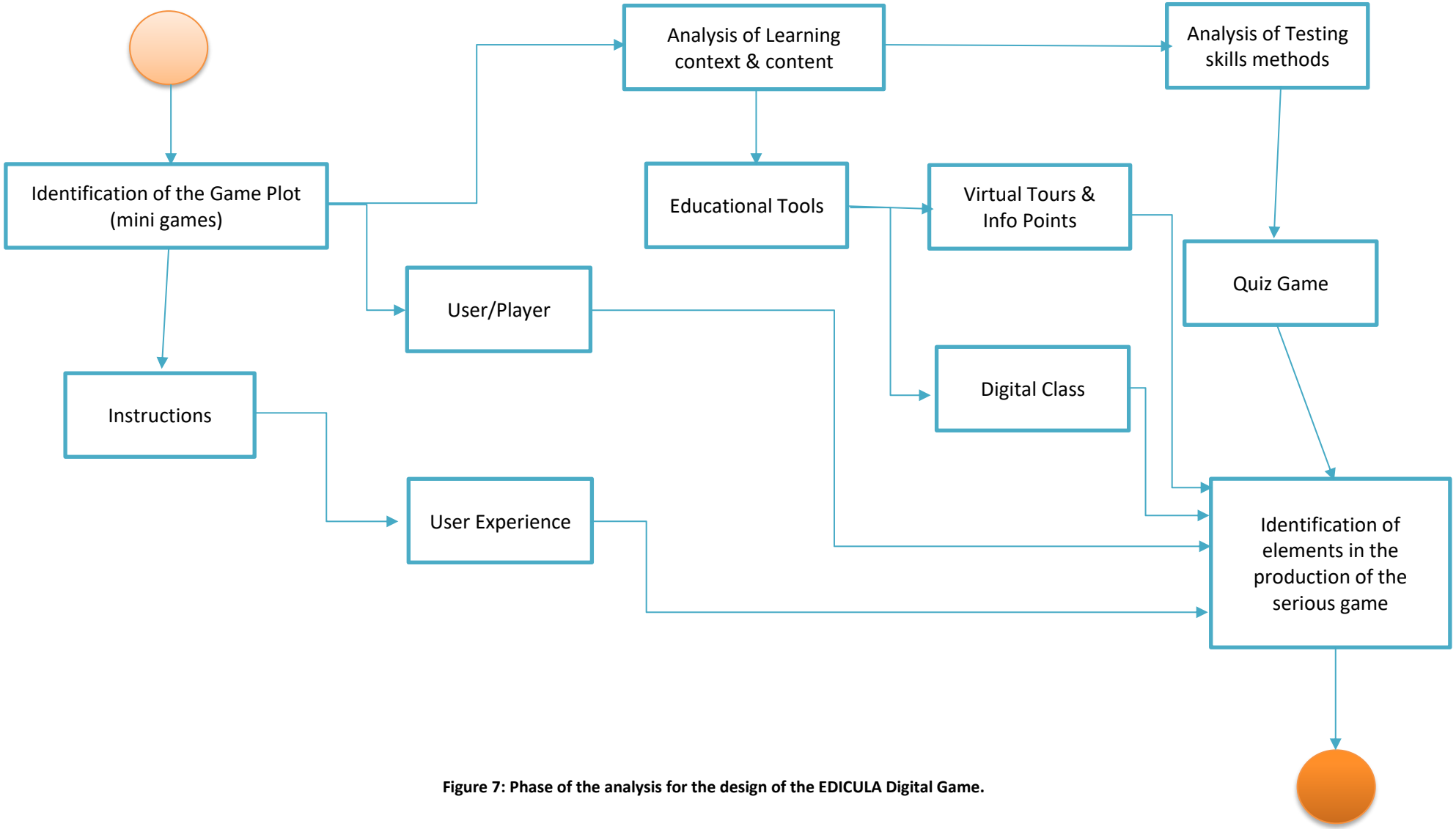

Figure 7: Phase of the analysis for the design of the EDICULA Digital Game.

4. Conclusion

Based on the research it is important to mention that choosing the appropriate game engine is not an easy task when developing a serious game. Moreover Unity3D is highly recommended for serious game developers both in the industry and in academia and will be used for the EDICULA Digital Game development. The architecture and analysis of the game design has been determined and the next steps will be the implementation and development.

REFERENCES

Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison Wesley, Boston (2003).

Bhargav, Vinay & Uskov, Alexander. (2015). Serious Game Engines: Analysis and Applications. 10.1109/EIT.2015.7293381.

Cooper, K.M.L., Scacchi, W.: Introducing computer games and software engineering. In: Cooper, K.M.L., Scacchi, W. (eds.) Computer Games and Software Engineering, pp. 1–27. Chapman and Hall/CRC, Boca Raton (2015).

Cowan, Brent & Kapralos, Bill. (2014). A Survey of Frameworks and Game Engines for Serious Game Development. Proceedings - IEEE 14th International Conference on Advanced Learning Technologies, ICALT 2014. 662-664. 10.1109/ICALT.2014.194.

Kruchten, P.B.: The 4+1 view model of architecture. IEEE Softw. 12, 42–50 (1995).

Masuch, M., Abbadi, M., Konert, J., Streicher, A., Söbke, H., Dey, R.: Lecture “Serious Game Technology”. In: Dagstuhl GI Seminar 15283 on Entertainment Computing and Serious Games (2015).

Murphy-Hill, E., Zimmermann, T., Nagappan, N.: Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development? In: 36th International Conference on Software Engineering (ACM), pp. 1–11 (2014).

Söbke, H., Streicher, A. (2016). Serious Games Architectures and Engines. 10.1007/978-3-319-46152-6_7.

Uskov, A., Sekar, B., 2014 - 2014. Serious games, gamification and game engines to support framework activities in engineering: Case studies, analysis, classifications and outcomes, in *IEEE International Conference on Electro/Information Technology*, IEEE, p. 618.

Xie, J., 2011. The research on mobile game engine, in *Proceedings of 2011 International Conference on Image Analysis and Signal Processing: October 21-23, 2011, Wuhan, China*, IEEE Press, [Piscataway, N.J.], p. 635.

Zarco, Liliana & Siegert, Jörg & Schlegel, Thilo & Bauernhansl, Thomas. (2021). Scope and delimitation of game engine simulations for ultra-flexible production environments. Procedia CIRP. 104. 792-797. 10.1016/j.procir.2021.11.133.