THIMEL-CONTENT: AN INCLUSIVE CONTENT CREATION, GAME CUSTOMIZATION AND GAMEPLAY PERSONALIZATION TOOL

by

KRISTOFER ANASTASIOS BARIANOS

Informatics Engineer, Technological Education Institute of Crete, 2018

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SCHOOL OF ENGINEERING

HELLENIC MEDITERRANEAN UNIVERSITY

2021

Approved by:

Major Professor
Nikolaos Vidakis

Copyright

Abstract

In an educational context, offered information is of uttermost importance. School books have been meticulously designed for years before publishing. However necessary, the result is static, often outdated, information, that limits the possibilities and the educational value. The case is even more complex for Serious Games, where teams comprised of multiple professions and backgrounds collaborate to produce a single game, but the result must be published in a matter of months, not years, in order to be relevant. Concurrently, the gaming industry is experiencing unprecedented technological and conceptual changes, while the educational field is also transforming to include new, innovative, pedagogical views and respond to the modern reality. Nevertheless, educational games often fall a few steps behind in both fields. While dynamic content for games has long been a reality, and quite a popular feature of commercial games, educational games have not yet incorporated such features in large scales. Thus, Serious Games can and should provide a canvas for dynamic education content, attracting the interest of pupils and elevating the educational process. Yet, even in commercial games with immense funding, there are usually no customization tools provided with the game, thus the content is simply a selection of predefined possibilities, not truly dynamic. In this endeavor, we have reviewed the current state of dynamic content and systems for content creation, storage, and management. Through this exploration we uncovered valuable and relevant knowledge and experience in the field, allowing us to better understand the current shortcomings and needs. As a result of this insight, we have designed a proposed framework to elevate the educational value of serious games, through the active inclusion of educators in content creation, game customization and gameplay personalization. Additionally, we created a pilot implementation, proving the possibilities this framework would unravel.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to Dr. Nikolaos Vidakis, my supervisor, for his valuable guidance, trust and understanding during the long procedure of writing this thesis. I would also like to warmly thank all team members from NILE Lab for their help and collaboration, and especially Alexis Papadakis, Sekellaris Sfakiotakis and Spyros Bartokaymenos. Lastly, I thank my family and close friends for providing perspective, encouragement, and constant moral support.

Chapter 1 - Introduction

The term Serious Games was first introduced by Clark Abt in 1970[1]. Despite the colossal growth of the field, especially during the last couple of decades, the definition provided by Abt is still accepted as the core of serious games [2], [3]. According to this definition, a serious game is specified by a decisive educational goal that is absent in games where entertainment is the only priority and sole purpose. Despite the initial criticism video games received as a new medium, scientists and educators has reach a consensus on the potential and value serious games carry as an educational tool, supported by evidence of the contribution to education [4], [5], [6]. This has shifted the acceptance balance in favor of educational games, creating a colossal wave of development and utilization. Designing and developing digital games however is an intricate procedure where many professionals are involved, and this is only amplified when education is an added expectation [7], [8], [9]. Thus, in this rising industry, the need for specifications and frameworks was realized rather quickly. Many frameworks have emerged in recent years, in an attempt to solve various problems and to create roadmaps for successful education through gaming. Hanes and Stone [10], divide frameworks into two subgroups, those that are conceptual, such as [9], [11], aiming to include pedagogical foundation in the games developed, and those that are directed towards the implementation, in the sense of techniques for design and evaluation of serious games [12], [11], mechanics within games [4], [13], or even learning analytics [14] and other highly specific techniques utilized in serious games and learning. Apart from all these frameworks, authoring tools and environments have become popular with numerous options available. Such platforms enable educators, and other professionals, to create their own simple games, or other interactive media, for their students without the need for technical knowledge and background [15]. The problem with authoring tools is that educators work alone. They do create quality educational content, but unfortunately within a very limited frame, compared to what a team of educators, game designers and developers could contribute to the learning process [9].

Serious games have grown and expanded to a point where features are rich within individual games and highly varied across different ones, candidly affecting the learning experience. The significance of the elements within a game and the way in which they affect the experience must therefore be understood, if we wish to create better circumstances for learning

efficiency [4]. However, as already understood, serious games are first and foremost concerned with educational goals. While game constituents significantly influence the results, the main element of any educational game is the educational content and learning components, in any form they exist and interweave with the gameplay. The challenge with serious games is to have gameplay and learning coincide seamlessly [16]. However most proposed frameworks, as well as released serious games, heavily lean towards one end, neglecting the other, possibly due to the difficulty of unifying said ends, limited timeframes and lack of specialization [17], [18]. In this context, serious games design and development is highly fragmented, with eminently diversified ambitions and pursuits. In some perspectives, this is a privilege, since it creates broader research that will result in better comprehension in the long run. Yet, this situation is unfavorable for present time developments, as the far-reaching spectrum cannot be adequately met. Additionally, resources are spent on recurring tasks and solving problems already solved in other contexts, hindering the progression of serious games and incorporation of new ideas, for instance the incorporation of emotional elements to facilitate recollection, as proposed by Malliarakis et all [11]. This results in a focus on highly abstracted goals, in detriment of tangible elements, considered non-challenging. Educational content often falls within this range of unremarkable elements. Consequently, it is given low priority within game design and development and is often approached from an inadequate perspective or is simply overlooked. However, it can be argued that content is one of the most important elements within an educational game. Thus, more attention must be put towards creation of meaningful educational content, as well as the fusion of educational material and game constituents, for any and every produced serious game.

Additionally, the education fields is growingly incorporating new techniques of adaptive learning in an attempt to create personalized learning [19]. Those techniques are applied to eLearning platforms, adjusting elements such as content presentation or navigation. Thus, a situation arises where learners in the same class doesn’t follow a preset learning path, but get to experience a unique educational situation [20]. It is shown that this approach motivates students to raise their engagement and better absorb information, leading to higher levels of learning [21]. In many cases, the gaming industry has employed techniques for personalization of the gameplay, even though with different goals. Despite some research and attempts done in serious games, those endevors generally explore possibilities and edge-cases while trying to utilize techniques, such as fuzzy logic [22], in new ways. However, those ventures have not resulted in

actual application, while serious games and game based learning are still very sparingly used in official educational context [23], [24]. Thus, the opportunity arises to utilize current technologies to create serious games that can adapt to students. The goal here should be to facilitate the incorporation of serious games into official educational structures, such as classrooms.

Based on the above observations, we propose a framework that will hopefully solve some crucial issues and facilitate the creation and utilization of educational games with increased educational value. This framework is based on a three-fold of principles, asynchronous multi-discipline collaboration, reusability and adaptive content. Following this model, professionals from the gaming industry and pedagogical fields should collaborate towards making games with unlimited potential. Additionally, games should be reusable, maximizing the time spent playing per time spent creating for each game and finally, educational content must be adaptable and of central importance for those games. Through this three-fold, educators can modify the educational content of games, making them reusable and customized to their students' needs, thus achieving all three goals. It brings power to educators, in a similar way to authoring tools, while keeping designers and developers in the loop. Thus, produced games are of higher quality, technologically relevant and more attractive to students who can enjoy the learning procedure and gain additional education value from it.

Chapter 2 - Background

Despite the fact that other industries often overlook and disregard video games due to their playful nature, they are one of the most sophisticated products in the modern marketplace. To create games a long list of interdisciplinary professionals is required and a multitude of separate technologies are utilized. When it comes to serious games, the list grows even more, with additional professions, technologies and tools that bring the educational/training element to the effort. This interdisciplinary state brings a lot of terms to the table, often relevant only to one of the contributing fields. However, in many cases these concepts can be utilized in other fields as well and bring innovation.

Reusability & Learning Objects

Software developers have been reusing existing software, and knowledge, to develop new solutions since the dawn of computer programming [25]. It has always been a goal to use existing work and experience to avoid spending time and resources on tasks and problems already solved, but also to ensure better software quality [26]. In today's software industry, that is overflowing with frameworks and nimble projects, software reuse is highly practiced. In the gaming industry aside from code reuse, such as inheritance and frameworks, there are complete game engines that offer a large amount of groundwork for reuse. Additionally, numerous shops offer game assets for sale, including 3D models, implementations for specific tasks, media, user interface graphics and complete packs including any combination of solutions [18], [27]. Serious games are generally created using such game engines and utilize shared assets when suitable. However, serious games also include elements specific to education purposes, such as assessment modules, learning metrics and the education content, that are not commonly shared in this manner. This is a problem recognized in the community of serious games. As a fact, the Games and Learning Alliance [28] identified reusability, and many aspects of it, as a prominent research and development challenge, thus deciding to include many of those aspect in its roadmap for non-leisure games. On the education side, Dawnes [29] realized the need for reusable blocks of educational material he named Learning Objects, and proposed them in 2001. Dawnes foresaw the digitization of learning and recommended reusability of elements common

through curricula. In his vision, instead of spending resources to produce numerous versions of the same material we could reuse the existing versions saving time and achieving better quality on each individual piece of material. The term has since been abandoned to a large extend, but his idea is partially put to practice through open resource, creative commons and other sharing protocols and structures.

For the technical aspect, there have been attempts, with various levels of results, to create structures for sharing, locating, reusing and customizing elements. However, most of them are proposed reference systems, with no or little concern for the entirety of the creation and reusability process. Most prominent are the efforts of the RAGE project [30] and the Serious Games Society [31]. RAGE aims to create assimilative, reusable and portable assets specific to serious games. It relies on latest technology, hoping to decrease investments and development time required by small studios, while increasing the quality of produced serious games [27]. The Serious Game Society has produced a number of tools and systems aiming to facilitate serious games creation, including a catalogue of web services which is now abandoned, a framework for learning analytics, called GLEANER, and a reference system for the identification of assets shared in the community [32]. GLEANER has been incorporated and further developed by RAGE project, becoming RAGE Analytics [33]. The latter of the societies efforts, namely Serious Games Reusability Point of Reference [34], is an effort towards the reusability of assets for serious games, where educational content could also be shared if properly modeled. However, it is simply a catalogue of assets provided by members, with no quality, portability or any other guarantees, without any means to assist developers adopt the available assets and without an established specification for the creation and submission of assets. For those reasons it never gained much traction. We can safely assume that RAGE is arguably the most unified and complete proposal for meaningful reusability within serious games. For that reason, it is also the framework that has made the largest contribution to the field. However, while the framework offers solutions to complex developmental problems, it completely excludes the most fundamental thing about serious games, knowledge. Educational material is in no way considered an asset by the RAGE framework, possibly because producing content is not a task of technical nature, however reusability criteria do apply and such an inclusion would offer new possibilities to serious game developers.

Dynamic Content

The fact that each student learns in a different way [35], has led to numerus techniques for personalized learning. Adaptive Hypermedia has been identified since the early stages of internet to be a powerful educational tool that can provide personalized learning paths. The content is thus selected among available elements, dependent on selections that express the personal preferences and background of each student [36]. Therefore, the process respects diversity between students and supports the concept of inclusive learning [37]. Following this path, the next challenge is to create adaptive educational games. Techniques based on those of Adaptive Hypermedia have been experimented with in educational games. ALIGN was a quite successful attempt to introduce the adaptive nature to games. Despite its positive effect, ALIGN has a complex architecture that did not facilitate adoption by serious game developers, thus discouraging use. As a result, the creators concluded that accomplishing adaptive educational games through the Adaptive Hypermedia logic would require substantial financial and technical resources, thus impeding advances in the field [38]. As a result, further efforts are limited, and alternative techniques were introduced. One interesting approach is that of the IOLAOS platform [39]. In an attempt to minimize technical difficulties and offer portability, IOLAOS offers a number of web services, from which developers can retrieve specific information about the current player. Such information can include age, preferred learning style, possible learning difficulties, etc. It is then up to the design and development team of the game to decide to which extent the game will be adaptable and how this information is going to affect gameplay and narrative [9], [23]. This level of freedom is supported by other factors in the platform and is justified on the premise of portability and creative flexibility. Nevertheless, despite the multitude of offered services, no tools or specifications are offered to assist developers and educators create and manage reusable educational content. Once again, developers need to design content management systems and seek collaborators to device educational content for their games. Additionally, a problem shared throughout techniques, is that content is not truly dynamic. It can be adaptive, usually based on “dynamic” content chains [40], however those chains are dynamically constructed, but with a predefined, limited selection of links that cannot be enriched without the active participation of the development team and the release of a update.

Existing Work

3D Repo

3D Repo [41], [42] is a version control solution for building information modeling. Version control, by definition, being a collaboration tool, 3D Repo enables multiple stakeholders to share a single repository and work on the same 3D asset, having a unique source of truth and simplifying the exchange of versions and 3D data. As depicted in **Figure 2.1 - 3D Assets exchange without and with 3D Repo [42]**, there are multiple transactions regarding information between different professions involved in the process of building construction, maintenance and demolition. However, with 3D Repo, all those exchanges are replaced by a single centralized source that each profession uses individually. Additionally, the system has integrated a REST API, facilitating interaction with the servers and allowing a multitude of integrated solutions to be created, and support for X3DOM [43], a HTLM based 3D integration. 3D Repo is available as a Software as A Service through 3drepo.io [44]. Along many more additions and features 3D Repo has released, is a library for Unity3D [45], that allows unity to load assets from the 3D Repo servers into the Unity3D engine at runtime. This is particularly useful in the constructions industry, as many representations utilize Unity3D. This particular library is implemented using .NET, not Unity itself, with the intent of reuse in other circumstances. However, some specifics, like the difficulties in asynchronous loading in Unity, and the way 3D models are rendered, makes a big part of the library specific to Unity. The result is quite interesting, the models are gradually imported, mesh by mesh and later textures are applied. However, there are difficulties in manipulating the created *GameObject*, as each mesh is imported as a child game object. In the context of construction industry, 3D Repo is an excellent solution that solved a multitude of issues, but unfortunately it is quite unfit for other fields due to its high-level of specialization and overhead. However, being opensource, parts of it could be extracted and repurposed for other more general frameworks, or even frameworks specific to other usages.

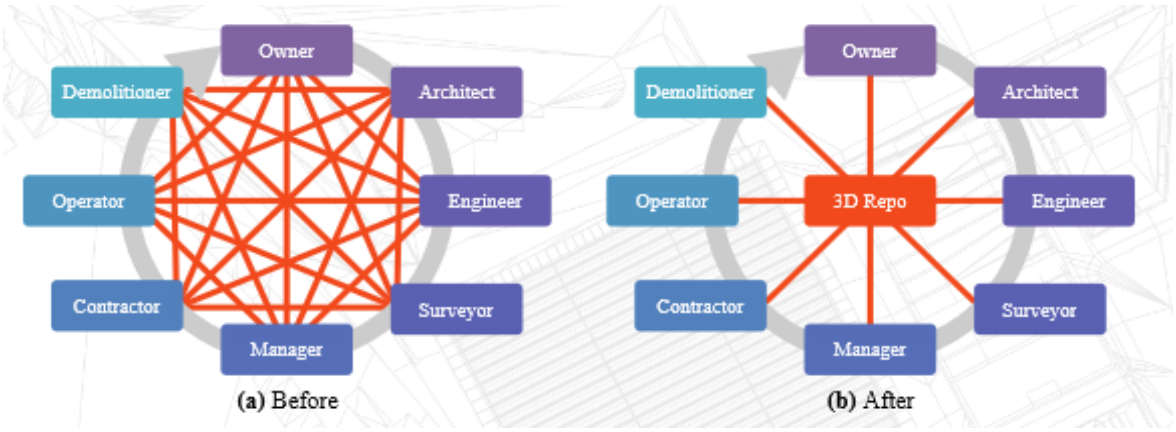

Figure 2.1 - 3D Assets exchange without and with 3D Repo [42]

DynaMus

DynaMus [46] is a dynamic 3D and 2D virtual museum framework, built with Unity3D real-time engine. In many ways, the framework can be considered as a 3D content management system. It allows users to browse museums created by other users, and build their own, either in 2D or 3D format. The framework is built using a server-client architecture, as illustrated in **Figure 2.2 - DynaMus Architecture[46],** where the client is the museum created with Unity3D, and the server is an application built in PHP that can locate images and 3D objects from URLs. Communication between the two is achieved with JSON data-interchange format. What is interesting is that the data is usually not stored on system servers, instead the Europeana web services and Google web services are utilized.

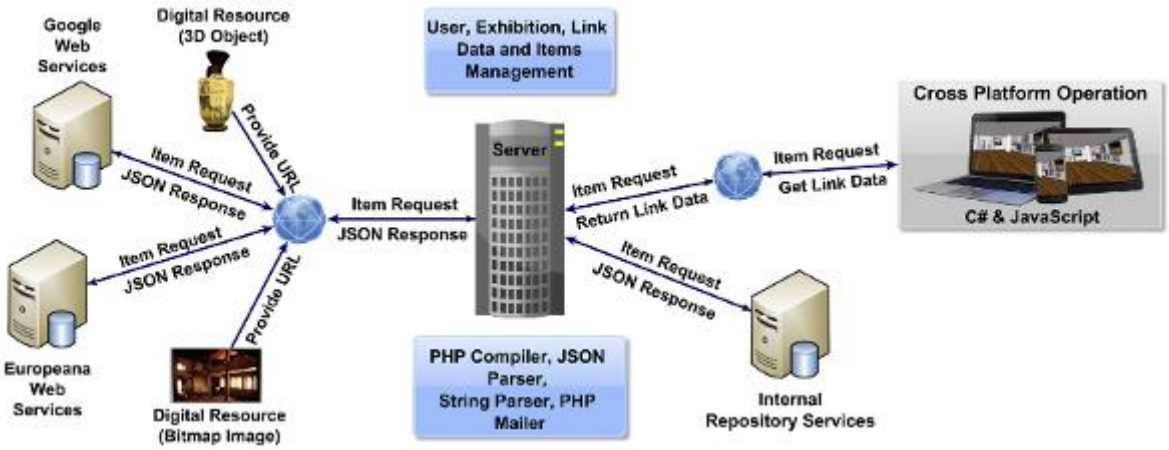

Figure 2.2 - DynaMus Architecture[46]

Comparison

Though quite different, both presented platforms provide the technical ability to import material into pre-compiled environments. DynaMus allows users to import images and 3D objects into a precompiled virtual museum, effectively changing what is displayed in the museum, while 3D repo offers, among other services, the infrastructure to import 3D objects into Unity3D projects, with the goal of showcasing and studying BIM data in a virtual 3D space. In this regard, DynaMus is useful to educators that wish to present virtual material in a suitable pre-defined environment to their pupils, while 3D Repo is useful to virtual environment developers who can create environments in which 3D buildings can be loaded post-compile. Neither of these is a complete solution, that can allow game creators to build customizable games, and allow educators, with no technical knowledge, to modify the games and create an experience suitable for their students. In further analysis, **Table 2.1 - Existing Work Comparison** offers some insight. The only common trait is that both technologies utilize a web architecture to store and retrieve the content. 3D Repo additionally doubles as a version control system for BIM data, while DynaMus does not offer any collaboration capabilities. 3D repo also provides a Unity3D Plugin to utilize BIM data stored on their servers through Unity3D engine. Both provide tools to change 3D models, but DynaMus offers the same capability for images and descriptions as well. The main usage of 3D Repo is commercial, building industry, creating showcases and environments for technical collaboration, while DynaMus targets education, creating a virtual museum environment. Lastly, though both have technical innovations useful in serious games customization, neither of them is developed in a way that could be applied to serious games, instead only parts of their systems can be repurposed towards such a goal.

Table 2.1 - Existing Work Comparison

Traits	3D Repo	DynaMus
Web Architecture	Yes	Yes
Version Control	Yes	No
Unity3D Plugin	Yes	No
Content Modified	3D Models (BIM data)	Images, Description, 3D Models
Main Usage	Commercial	Educational
Virtual Education Environment	No	Yes
Applies to serious games	No	No

Our Proposal

Our research of the field led us to the understanding that serious game content is underrepresented in research and developmental efforts. The technological reality of the day has lots to offer towards quality content, improved methods for creation, management and collaboration, but also elevated levels of freedom to each professional involved in serious gaming. What we propose, as a means to assist in the investigation of the research question and as groundwork for future endeavors, is a framework to facilitate dynamic educational material management and utilization within serious games. This framework will hopefully bring truly dynamic content to serious games and will improve material through a different approach to creation, that is detached from the initial design and development of the game. To analyze, we share the aforementioned assumption by Vidakis and Charitakis [9] that each professional has highly specialized skills that cannot be overlooked or replaced. Authoring Tools, though great means for educators to create interactive material, create very limited experiences compared to games with the level of attraction that gaming studios publish, while at the same time, gaming studios cannot create and include educational material of quality, personalized to the last class and student, even with tight collaboration with educators, educational experts and authors [9]. However, it is obvious that such collaborations, as all collaborations, are time and resource demanding. In the age of computer-supported collaborations, any interdisciplinary field should effectively utilize technology to create better circumstances of collaboration that demand less from all parties while delivering superior results. Additionally, it is critical that content should be able to be enriched and modified without involving the developer and creating updates and new, specific, releases. Thus, what we envision is educational content that includes suitability metadata, complemented by various specifications and infrastructures to cater to the needs of all stakeholders. In support of this, a framework will specify rules and offer assets for integration into games, so that content can be dynamically coupled to the game, at run time. This coupling will depend upon game and player criteria, selecting only content that is suited to the game narrative and theme but also accords to the current player's needs. It will then be modularly added and displayed in game. This selection and retrieval process will be realized by web services communicating with a content repository. Additionally, infrastructure must be in place for educators to submit educational material in the form of game content and limit available content, or choose specifics, for their learning sessions with their students. This will be realized

through a web site, similar in functionality to modern Content Management Systems. A Problem that must be foreseen and addressed during development is that the web services make gaming sessions dependent on internet connectivity. This can be solved through “educational packets” that can be saved into a specified local folder that can be used when an internet connection is not available.

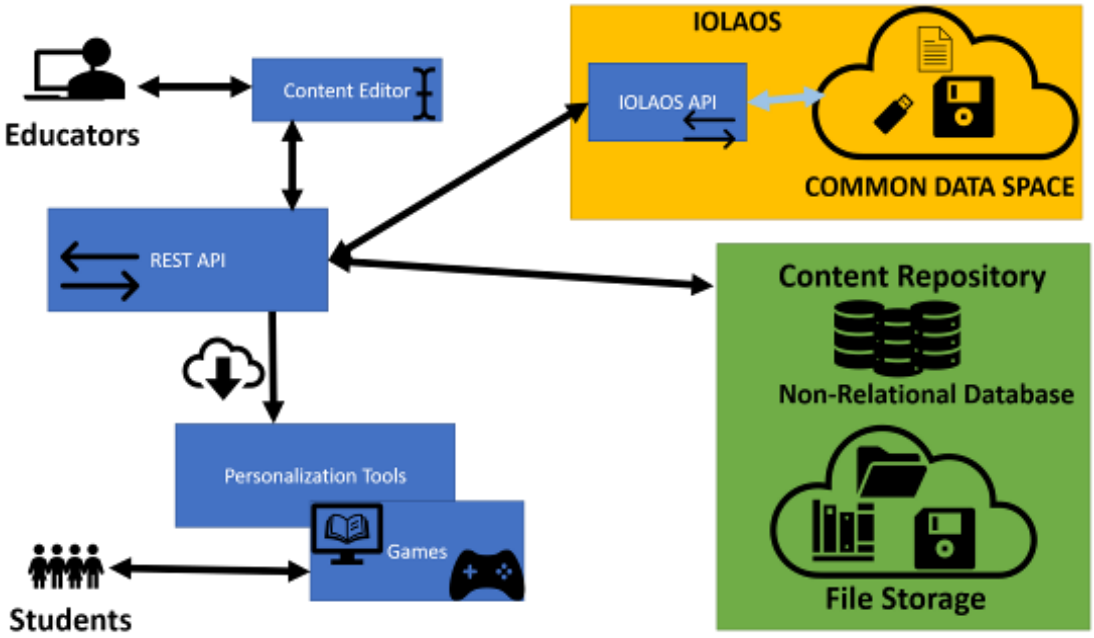

Figure 2.3 - System Overview

The overview is simplified and depicted in **Figure 2.3 - System Overview** as follows. Educators interact with a content editor to author new educational material or select material for an upcoming learning session. The content editor communicates with the RESTfull API to authenticate the educator and access/modify learning session data, through the IOLAOS API. It also communicates with the content repository to retrieve the available material and to save any new material submitted. Finally, the students access the game, which has personalization tools included, that will communicate with the REST API to authenticate students and retrieve the appropriate material for each student from the content Repository. Determining which material to retrieve will be an elaborate process, that will combine user profile (such as age and learning preferences) and learning session.

Functional and Non-Functional Requirements

From the goals stated above we can identify the requirements of the system, which were analyzed and composed into **Table 2.2 - Functional Requirements** and **Table 2.3 - Non-Functional Requirements.** The Functional requirements are grouped into 3 groups depending on what aspect they address, IOLAOS, Editor or Data Storage. These groups are also expressed through the ID names, where FI is Functional-IOLAOS, FE is Functional-Editor and FS is Functional-Storage.

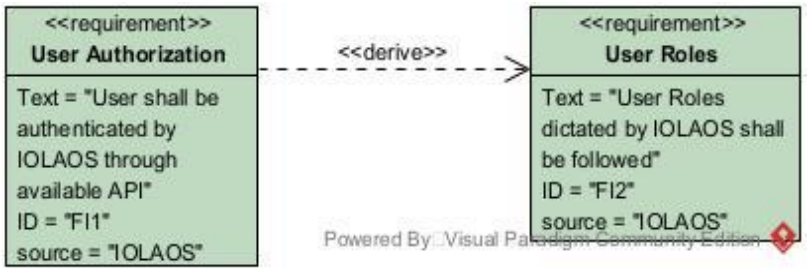

Figure 2.4 - IOLAOS Functional Requirements

Functional requirements regarding IOLAOS comprise the first group, depicted in **Figure 2.4 - IOLAOS Functional Requirements**, specifically there are 2 requirements to make the platform compatible with IOLAOS. Users shall be authorized through IOLAOS with the provided API (FI1), and user roles dictated by IOLAOS must thus also be followed (FI2). The second group is related to the Content Editor we aim to create. These requirements are illustrated in **Figure 2.5 - Content Editor Functional Requirements** in detail. A web page must be created that will be the Content Editor (FE1). Through this website, any existing content will we accessible for educators to view, however there will be access limitations (FE2). Additionally, the content editor will allow material creation (FE3) and editing (FE4). Additionally, game developers will be provided with the ability to register their game to the system (FE5) and will define the content structure for their games (FE6), in order to allow authors to create educational material for the game. Content will be accessible to the creator and his/her pupils through a token (FE7), while the official content (Content that is available to all users) cannot be deleted, instead, it can be archived (FE8). Lastly, the content editor must be multilingual, and as a minimum the initial version shall support Greek and English (FE9).

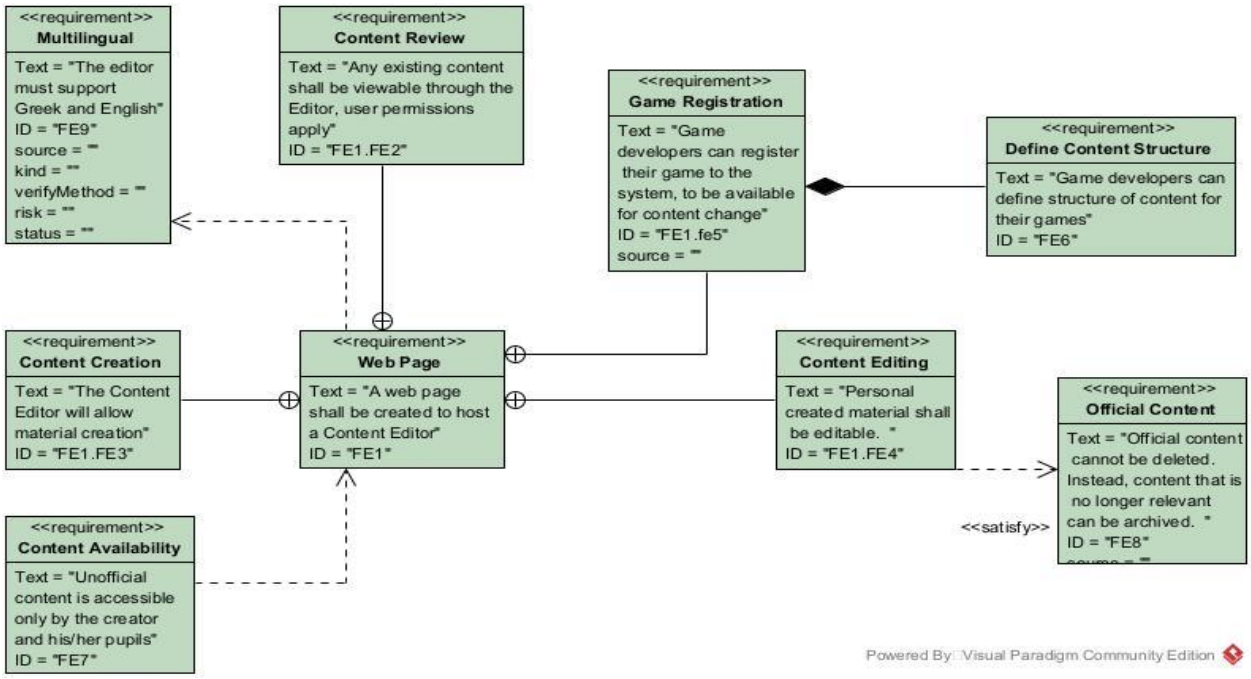

Figure 2.5 - Content Editor Functional Requirements

The third group, Data Storage is portrayed in Error! Reference source not found.. This g roup is concerned with data management. Specifically, the requirements of this group are four. Content shall be stored in remote servers (FS1), all data shall be retrievable from those servers by the editor and any games, with appropriate filters applied (FS2), media types stored on the servers shall include Text, Audio, Video, Images, and 3D models (FS3).

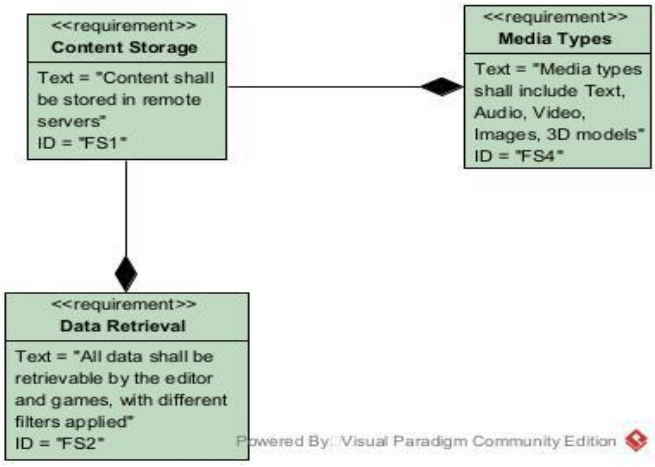

Figure 2.6 - Data Storage Functional Requirements

Table 2.2 - Functional Requirements

Functional Requirements		
ID	Title	Description
FI[1]1	User Auth	User shall be authenticated by IOLAOS through available API.
FI2	User Roles	User Roles dictated by IOLAOS shall be followed.
FE[2]1	Web Page	A web page shall be created to host a Content Editor.
FE2	Content review	Any existing content shall be viewable through the Editor, user permissions apply
FE3	Content Creation	The Content Editor will allow material creation
FE4	Content Editing	Personal created material shall be editable.
FE5	Game Registration	Game developers can register their game to the system, to be available for content change
FE6	Define Content Structure	Game developers can define structure of content for their games
FE7	Content availability	Unofficial content is accessible only by the creator and his/her pupils (through token)
FE8	Official Content	Official content cannot be deleted. Instead, content that is no longer relevant can be archived.
FE9	Multilingual	The editor must support Greek and English
FS[3]1	Content Storage	Content shall be stored in remote servers
FS2	Data Retrieval	All data shall be retrievable by the editor and games, with different filters applied
FS3	Media Types	Media types shall include Text, Audio, Video, Images, 3D models

[1] FI = Funcional Requirements regarding IOLAOS

[2] FE = Functioal Requirements regarding the Editor

[3] FS = Functional Requirements regarding data Storage

Table 2.3 - Non-Functional Requirements

Non-functional Requirements		
ID	Title	Description
NT1[4]	Front-End	The website shall be created with ReactJS
NT2	API	The API shall be created with Node.js
NT3	API key	The API shall share API keys with IOLAOS for game authentication
NT4	Database	The Database shall be a MongoDB
NT5	Game Engines	The system shall include the tools necessary for usage with Unity3D game engine.
NP1[5]	Data Delivery Times	Data delivery time between parts of the system shall be kept under 1 second
NP2	Data Processing	Data processing and response preparation shall be kept under 0.5 seconds.
NP3	Simultaneous Requests	The system shall be able to handle at least 1000 simultaneous requests from any part of the system (games, editor).
NS[6]1	Text & Files Storage	Various forms of media content shall be stored in remote servers in NoSQL format.
NS2	Meta-data	All content data will be supported by meta-data
NS3	Secure File types	File types shall only be supported if security against viruses can be assured

[4] nFRs regarding technologies to be used

[5] nFRs regarding performance

[6] nFRs regarding Storage

Non-functional Requirements are grouped into three groups as well, Technologies, Performance, and Storage. Technologies Non-Functional Requirements (nFR) are portrayed in **Figure 2.7 - Technologies Non-Functional Requirements**, and include the need for a ReactJS front-end website (NT1), an API built with Node.js (NT2) that will act as the middle-man between the games, the website and the database, a common API key with IOLAOS (NT3) for game authentication, and, lastly, a MongoDB database(NT4) where all content will be stored.

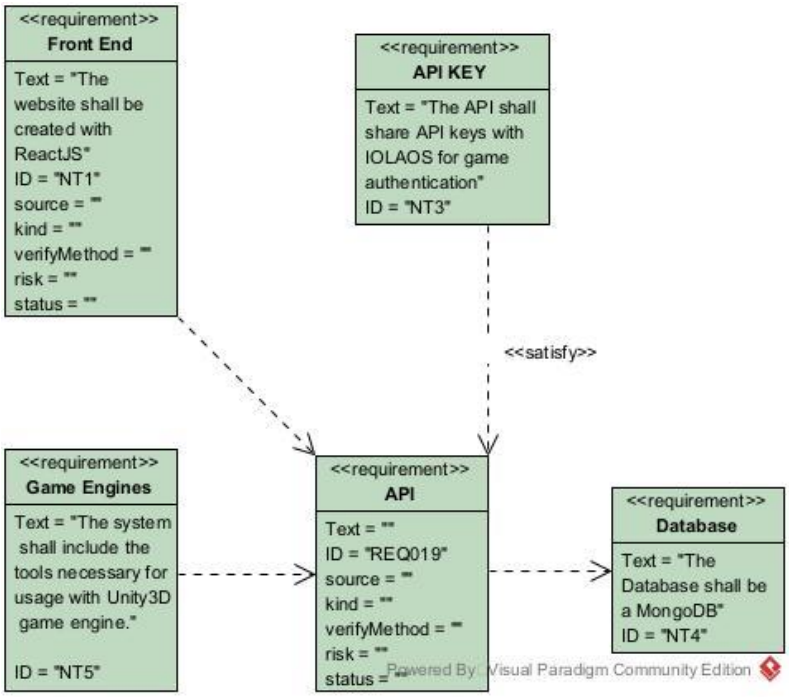

Figure 2.7 - Technologies Non-Functional Requirements

The second nFR group, performance, is depicted in **Figure 2.8 - Performance Non-Functional Requirements** and is concerned with performance and stability of the system. Specifically, this group defines the need for fast response times though two requirements, Data Delivery Times, which must be faster than one second (NP1) and Data Processing, which must be faster than 0.5 seconds (NP2). Additionally, this group defines the need stable infrastructure than can handle at least 1000 simultaneous requests (NP3), so that the system doesn't fail if multiple classes are trying to play games concurrently. The final nFR group, Storage, depicted in **Figure 2.9 - Storage Non-Functional Requirements,** defines the needs regarding data storage. Storage format shall be NoSQL in the remote servers (NS1), all data must be supported by meta-data (NS2) and lastly, only file types that can assure security shall be allowed (NS3).

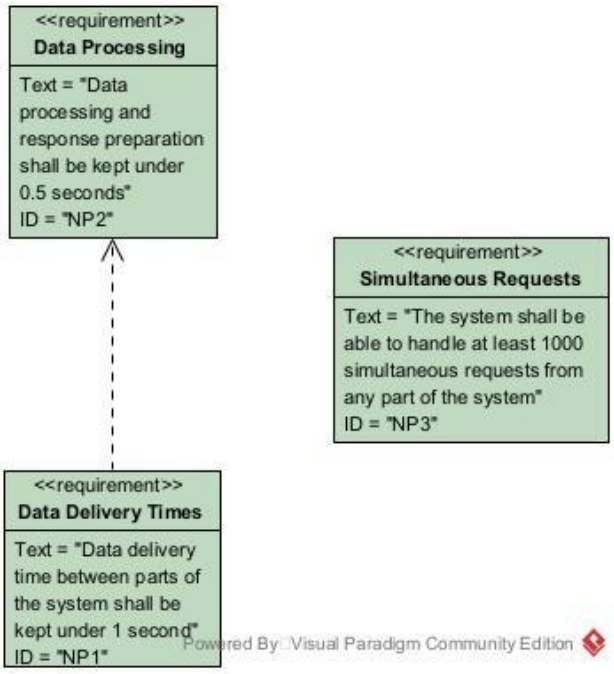

Figure 2.8 - Performance Non-Functional Requirements

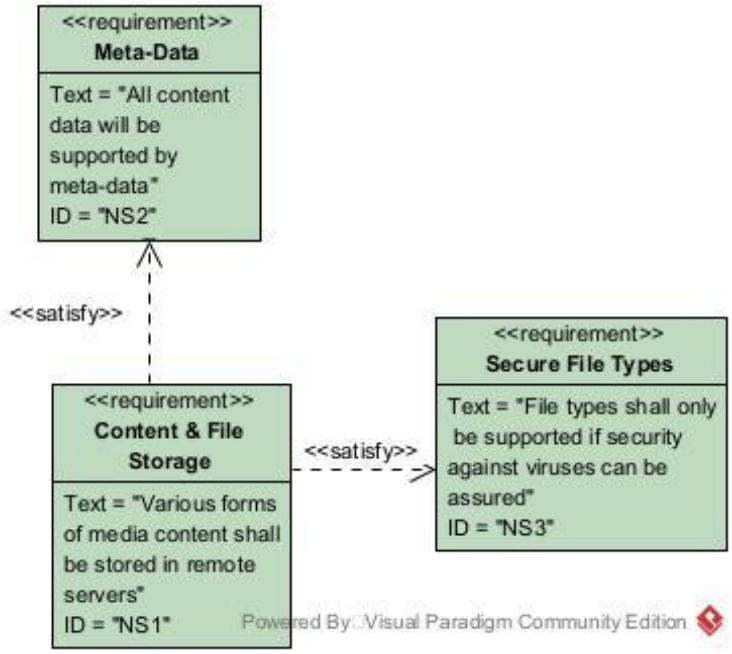

Figure 2.9 - Storage Non-Functional Requirements

User Roles

Table 2.4 - User Roles

User/Role	Example	Frequency of Use	Security/Access, Features Used	Access to
Game Developer	Game Developer of ThimelEdu	Rare or Occasional	1. Game Registration 2. Content Structure Definition 3. Content Publishing	Editor & Game
Educator	History Teacher	Frequent	1. Content Creation 2. Assign content to student/class	Editor & Game
Learner	Student/ end game user	Frequent	No direct features	Game
Educational Expert	Special Education Specialist	Inactive	No direct features	Game
Game Tester	Inactive	Inactive	No access	Game
Administrator	Admin	Frequent	Full Access	Editor

As described by the Functional Requirements stated above, the system will follow the user roles defined by IOLAOS. However, IOLAOS is a platform that covers a broader spectrum of educational needs compared to the proposed system, and as such has more roles than needed in this case. Therefore, some of the IOLAOS user roles will be inactive in the proposed platform, meaning that there will be no features designed for them. As already analyzed above, Game developers will be able to register their game with the platform, define the content structure and publish new content. Educators can use the platform to author new educational material or select custom material for their learning sessions. The last group already mentioned are the learners, who will access the platform indirectly, through the game, retrieving material that will customize their game and experience. The last role that will be able to use the platform is the administrator, who of course is a bureaucratic role, included only for maintenance needs, with no impact on the educational services of the platform. IOLAOS has two additional roles, Game Tester and

Educational Expert. Game Testers test games for compliance with “IOLAOS Maturity Levels”, which is irrelevant for this platform as the games will already be included in the IOLAOS platform, otherwise they can’t register with this platform. Additionally, even though IOLAOS might decide to make “Dynamic Content” an additional “Maturity Level”, this platform is only concerned with the management of educational material and the tools to bring personalization into games, testing for compliance is beyond our interest and scope.

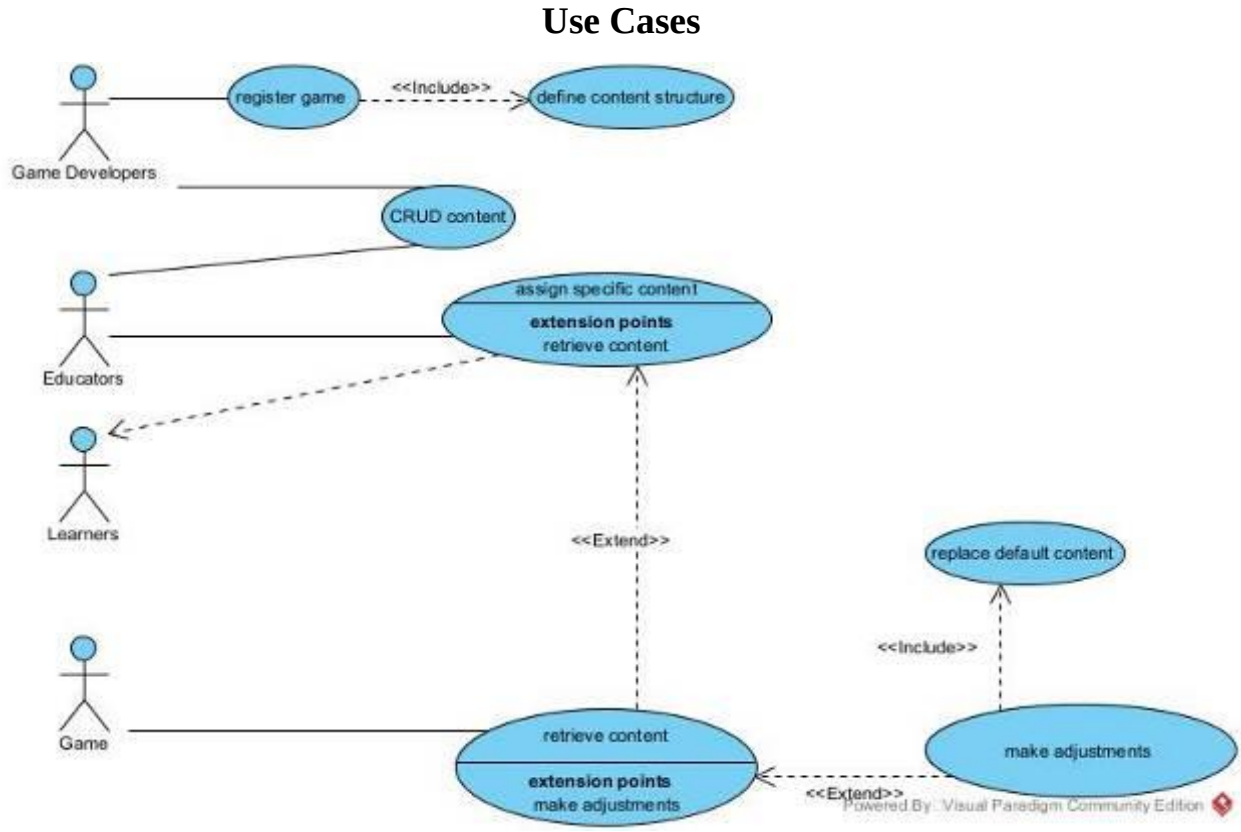

Figure 2.10 - System Overview Use Case

The requirements analyzed in section Functional and Non-Functional Requirements allow us to identify the use cases for each role involved in the process and better realize what actions are implied. Below, we will analyze the most important use cases of the system, namely:

- System Overview, Figure 2.10
- Define Content Structure, Figure 2.11
- Create-Read-Update-Delete (CRUD) Content, Figure 2.12
- Initialize Game, Figure 2.13

For the system overview, we have four actors, the three user roles that are actively engaged with the system, plus the games themselves. The chain of events starts with the Game Developers, who register their game with the system and define the content structure. After that, both developers and educators can use the editor to Create, Read, Update or Delete content. Educators can additionally assign specific content to learning sessions they teach, to specific learners. Lastly, games can retrieve content from the API, and make adjustments using the personalization tools, replacing the default content with material retrieved through the system. This brief overview is depicted in **Figure 2.10 - System Overview Use Case**. Further analyzing elements of this use case, we can identify the definition of content structure and the CRUD procedure as additional use cases. In **Figure 2.11 - Define Content Structure Use Case** we have illustrated the use case of creating the content definition for a game. To elaborate, the game developer will add elements, such as text, videos and images, and define the sequence in which they are to be expected.

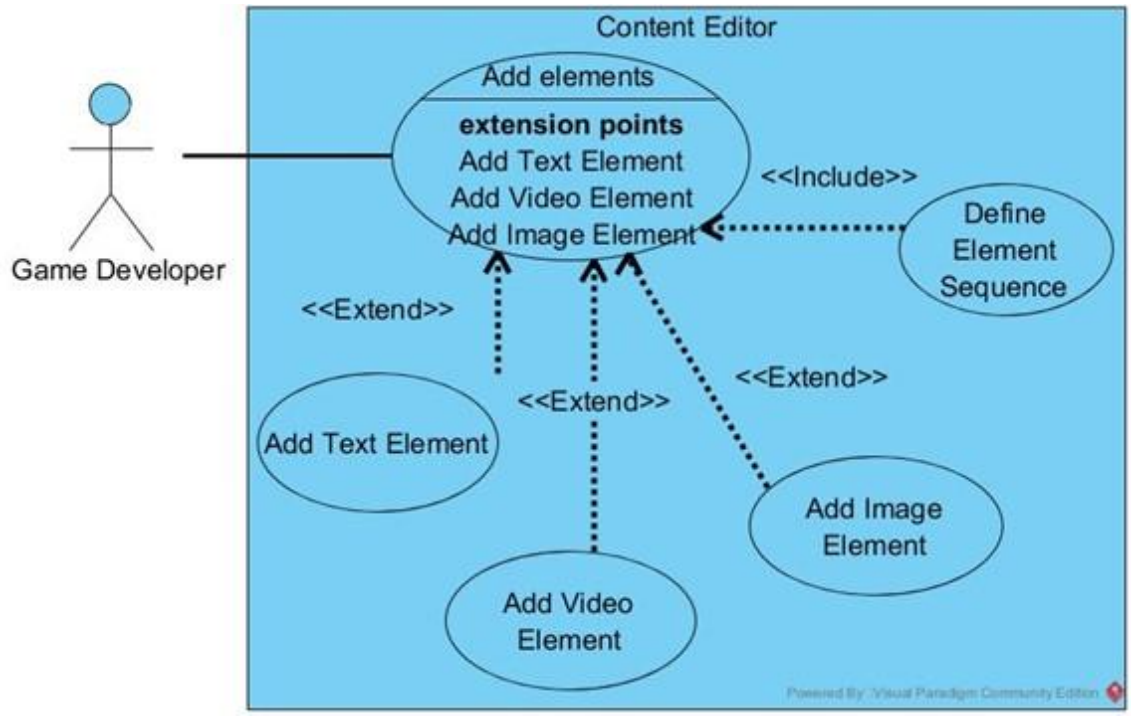

Figure 2.11 - Define Content Structure Use Case

Depicted in **Figure 2.12 CRUD Content Use Case**, we have game developers and educators as actors, but both roles can be seen as a single actor, an Author. This Author can

choose a game, for which the existing content will be displayed. The actor can now select of the available content packages, to either delete or edit, or to create content. Editing will imply editing texts and/or uploading media. Deleting will remove the entire content package from the remote server. Creating implies the authoring of texts and upload of new media. In either case, saving the changes is a necessary last step.

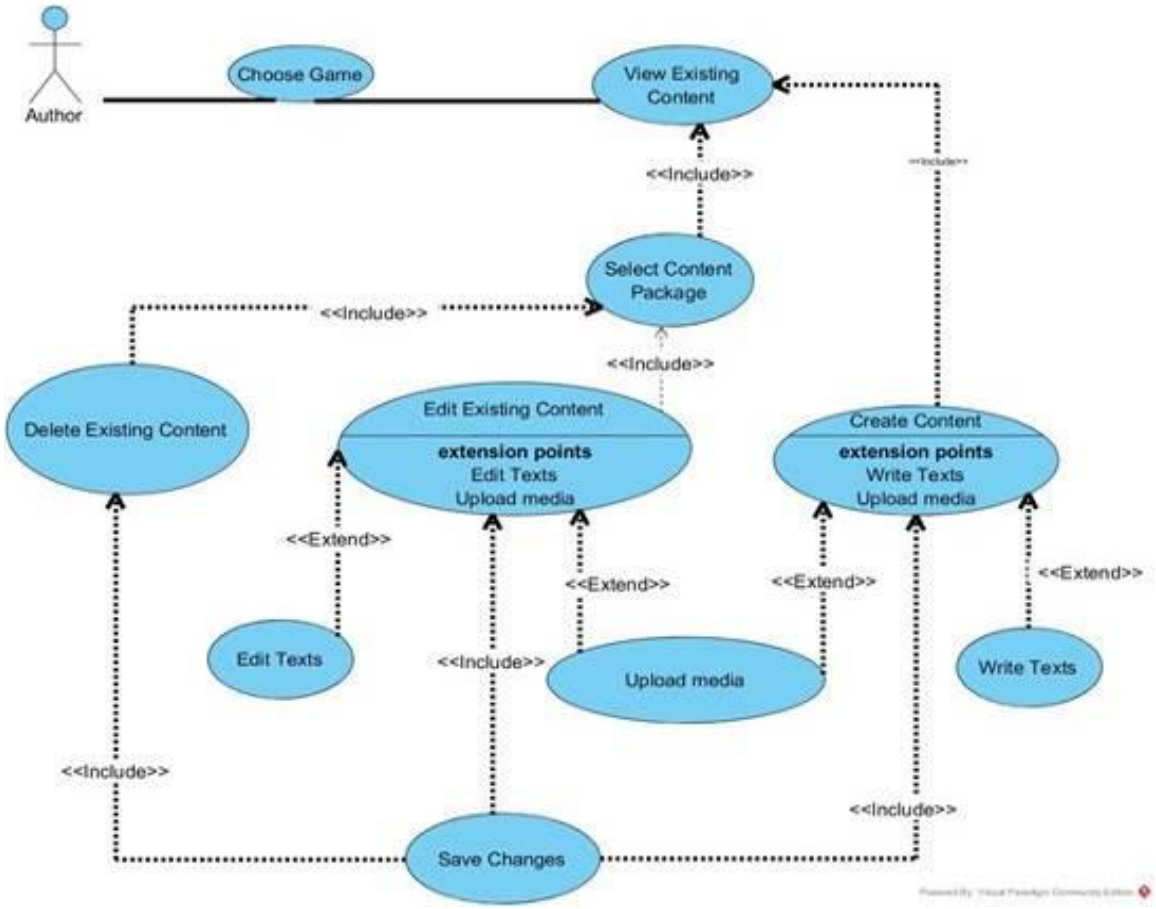

Figure 2.12 CRUD Content Use Case

Finally, the last use case we will analyze is depicted in Figure 2.13 - Initialize Game Use Case. For this situation, our actors are two, the Player and the game itself. The player, in this situation, might be a student but it is not certain. The educator might also join a game, and the games are also open to guests that are treated as simple players. Thus, in our use case we have chosen the most general depiction of the user. The player opens the game, from there he/she can either go on to play the game with the default settings, or login through IOLAOS to get a personalized

experience. Regardles of wheather the player will login, the game will do an initialization with default settings and content. Logging in will trigger two additional events, the game will retrieve session data, if there are, and the personal player preferences of the player. If session data is found, they are retrieved from the server and the applied to the game, updating the game content. This would also mean that our player falls in the category of student. The second action that is triggered is the retrieval of the player preferences stored in the user profile, that will in turn update the game settings to match this profile. Both of those updates will happen seamlessly, before the actual game play has stared. Lastly, the player gets to play the game, in whatever form is the final, either the default initialization, or the personalized version.

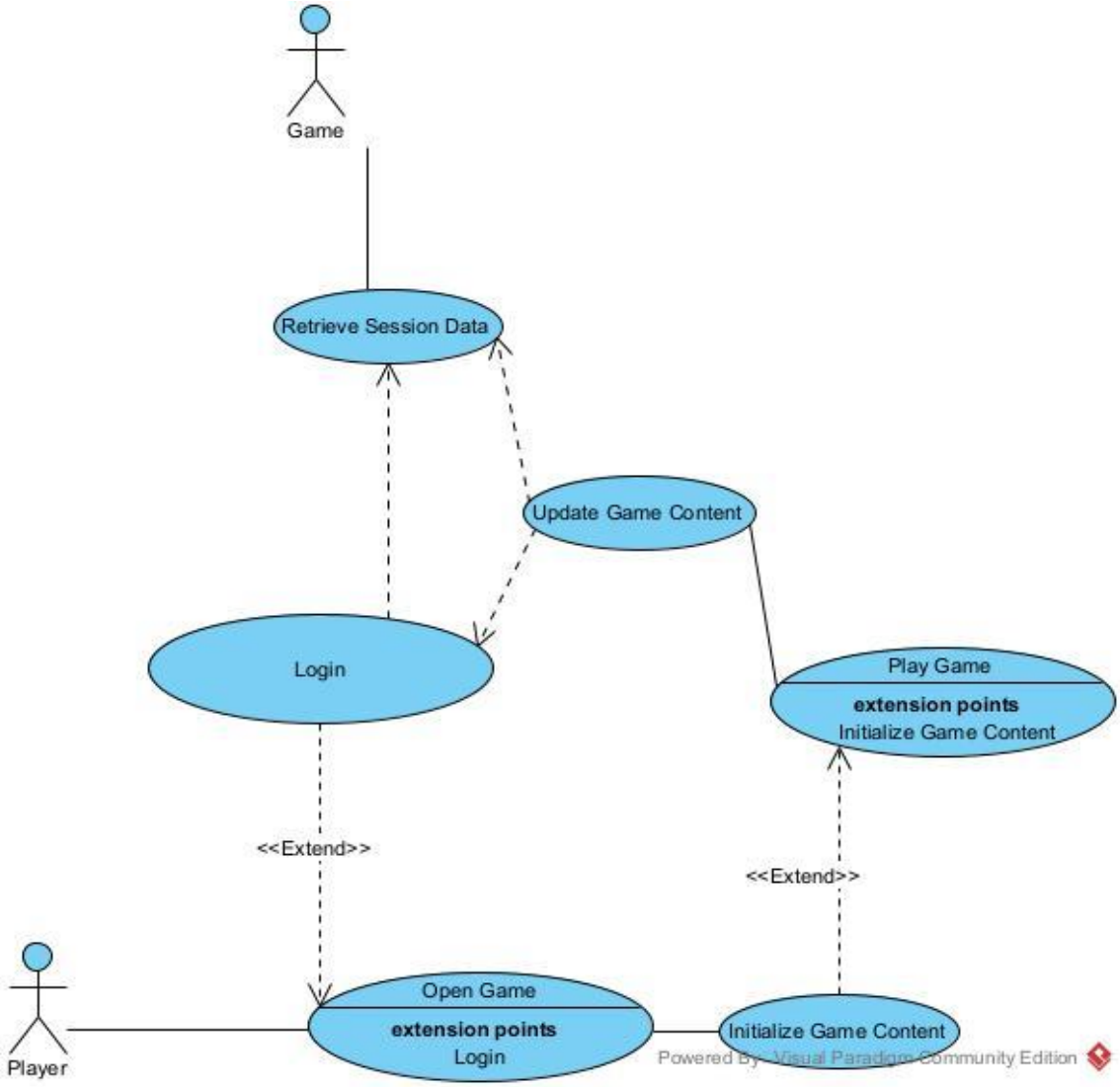

Figure 2.13 - Initialize Game Use Case

Architecture

From the analysis conducted above, we have arrived to the architecture of the proposed system and designed a component diagram, illustrated in **Figure 2.14 - Proposed Component Diagram**. Beginning from bottom right, we have a Data Storage, that will consist of a NoSQL database that will hold the content structure, texts, metadata and the locations of files, a File Server that will host the files needed for the content structure, and a Content module, that will be able to compose information from the database and file server into singe Content Objects. On the Upper Right, we have an API, the REST API will accept requests, the Data Filter will formulate the filters according to the specifications set by the request and the Data Manager will formulate queries to retrieve content or create and write into the Data Storage. On the upper left we have the content editor, a web site, which will include editing tools for authors to create or edit their material and will be able to visualize the content through its graphical interface. In the middle, we have the Dynamic Games, that will be able to request content through the API and personalize the game. Finally, on the bottom left, IOLAOS framework will accept request from the games and the content editor, to authenticate users, find learning sessions and classes, etc.

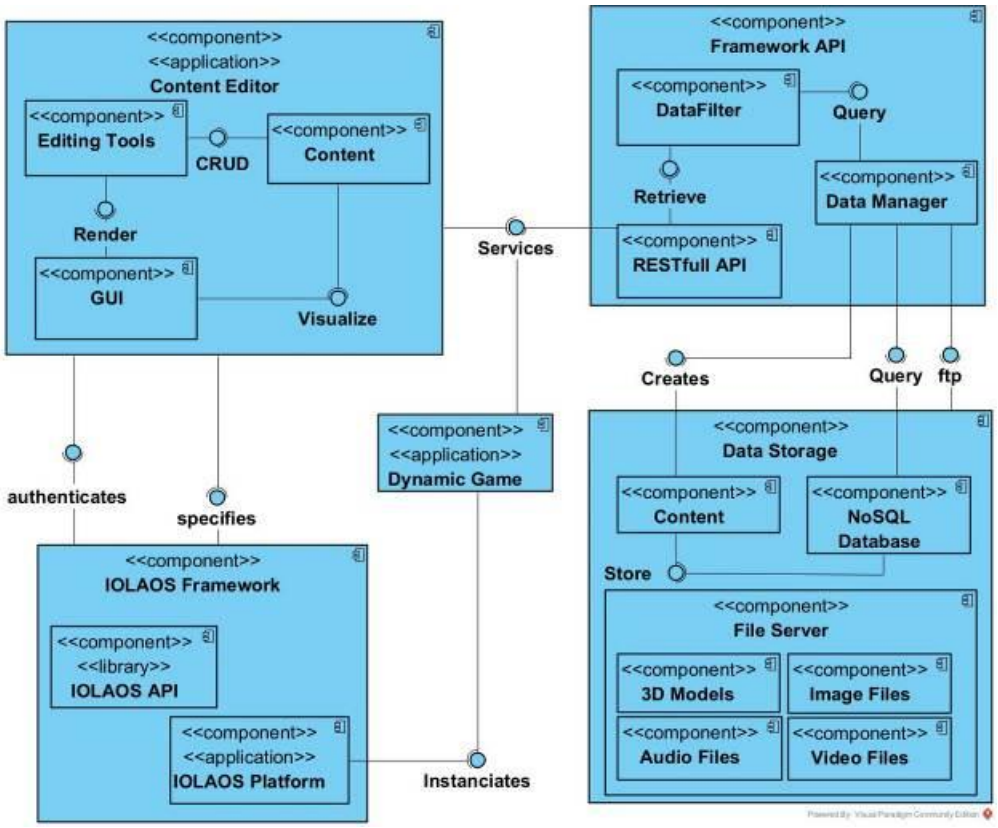

Figure 2.14 - Proposed Component Diagram

Chapter 3 - Implementation

For this thesis, we created a pilot implementation, with a portion of the architecture proposed above. We focused mostly on the techniques involved in making games adaptable at runtime and any inextricable modules. Thus, we identified the game adaptation engine, a database and file store, and the communication between them as the crucial parts for our pilot implementation. Creating a game adaptation engine is a very challenging task, as each game is different, and can be built upon a number of available game engines, a custom engine created by the developer, or even without a game engine, something that often happens with simpler web-based or mobile games, where a game engine would introduce a multitude of unnecessary features that have a cost in size and performance. Therefore, we have created an abstract architecture to guide all implementations, illustrated in **Figure 3.1 - Generalized Architecture**.

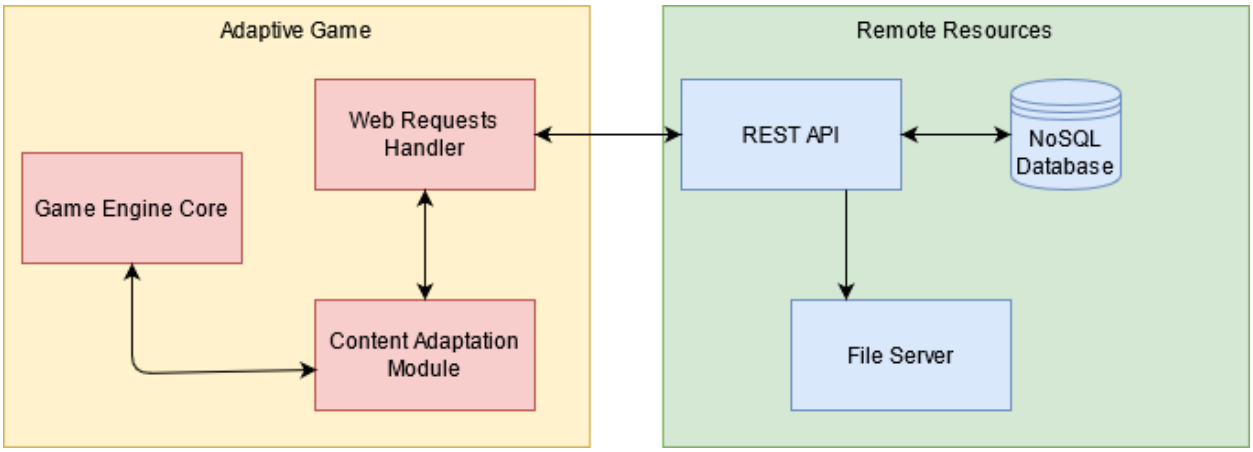

Figure 3.1 - Generalized Architecture

According to these guidelines, we expect a game engine core. This might represent something different for each game, as some will utilize their rendering engine to know what to render next, while others might have custom solutions to feed information to the rendering engine. In any case, in regards to this diagram, this represents any logic beyond the scope of our solution, be it game engine modules or customized code by game developers. Our Content Adaptation Module communicates with this core, getting insight as to what parts of the game need to be customized. Having this information, the Content Adaptation Module uses the Web Request Handler, which in many cases might be part of the game engine, to communicate with the outside world, in order to find the appropriate material for each adaptable element in game.

To support this, we need our Remote Resources, a REST API that can accept request, find the information needed and reply. The information will be stored either in a NoSQL database, or a file server, depending on the type of material and information.

For our implementation, we decided to realize the above architecture for one of the most popular game engines. Unity3D was selected, due to its popularity and previous experience of the author. Additionally, a serious game, ThimelEdu, was chosen to showcase the results in a real environment. This game was chosen as it already is IOLAOS certified, and the author is the lead developer of the game. Knowing that we will work with Unity3D real-time engine, we particularized the previous abstract architecture into a more specific component diagram, illustrated in **Figure 3.2 - Architecture for Unity Implementation**. In this case, the Web Request Module and the Content Adaptation Module, become part of the game engine core, as we wish to develop it within Unity, not as an external library. The only significant change is the addition of TriLib Library, due to the challenges of importing 3D models at run time. This is further elaborated bellow.

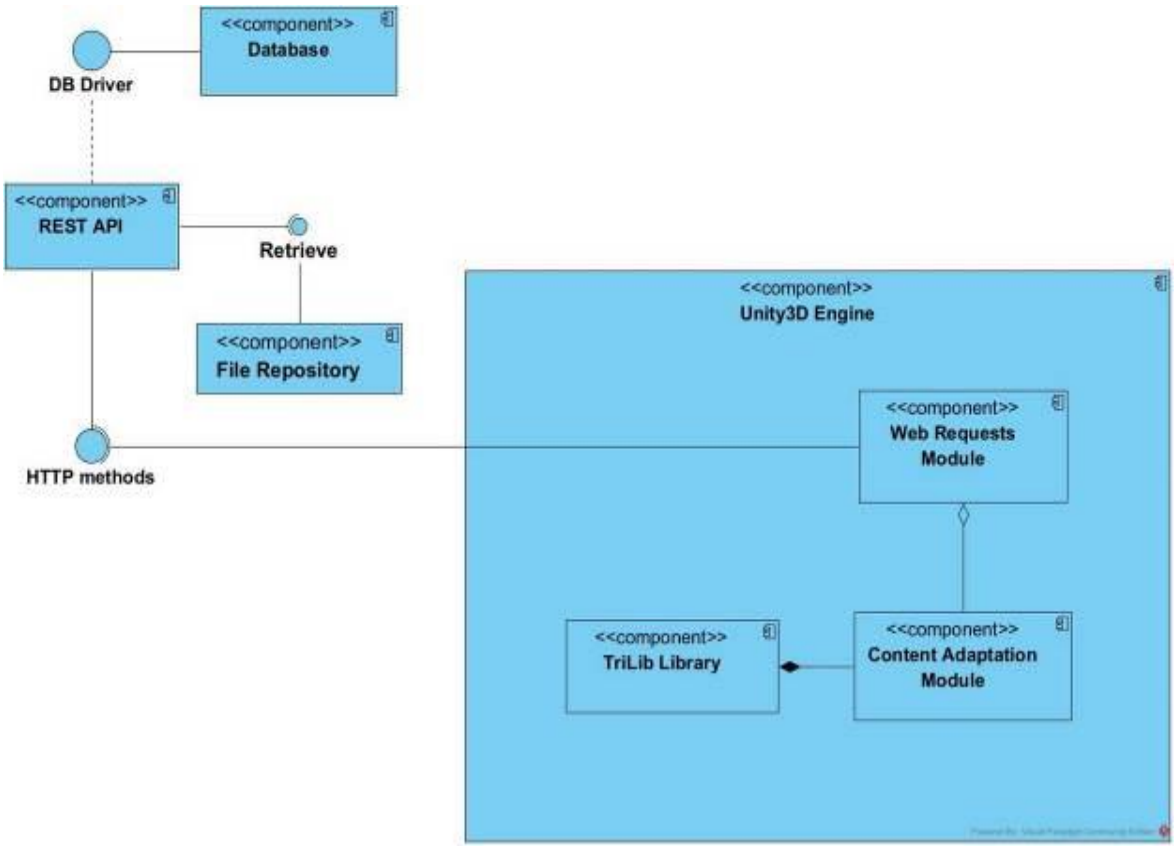

Figure 3.2 - Architecture for Unity Implementation

Technologies Used

ThimelEdu

To test and illustrate our implementation we used the interactive 3D serious game ThimelEdu [23]. ThimelEdu was designed and developed as an exploratory learning alternative, through which learners can discover and study educational material about the architectural elements and tools utilized in ancient Greek theatre. The specific topic is challenging for both teachers and pupils. For the former, recreating ancient environments and situations, in a manner that will trigger interest is a task with disproportionate difficulty compared to the time devoted to the topic in school syllabus. For the latter, studying such a unique topic from the static context of books and images is tiresome. Additionally, visits to such ancient sites, that would be the ideal teaching setting, is impossible in most cases, and difficult in the best cases. Therefore, the game aims to facilitate learning about the topic, as a complementary tool within the school environment [47]. In the game, a 3D representation of an ancient theater site is presented to players, who can navigate freely, in third person, through every part and discover architectural elements and tools. Players can interact with those 3D objects, and through them discover educational material, such as texts and images and answer quizzes. Scoring, rewards and other gamification elements are also included, to enhance players awareness, interest and performance [23]. The play sessions can also offer assessment, through the scores and quizzes though Learning Analytics [47]–[49][23]. Additionally, the game offers accessibility by adapting educational material and game settings, according to the current player. This is achieved through interoperability with the IOLAOS platform [9], ensuring the best possible learning environment and conditions for each and every student [50]. During gameplay, players are tasked with locating and identifying artefacts of ancient theater positioned throughout the 3D environment, interact with them in a pursuit of educational information. Additionally, after this interaction, quizzes appear to the player, offering real-time assessment and better retention of knowledge.

By Default, ThimelEdu comes with a selection of educational material. This material is characterized to be shown to the appropriate learners by language, age, and school type(as identified by IOLAOS). The entire selection is codified in JSON format in the Streaming Assets folder, so that it can easily be updated, even without the intervention of the game developer. In our case however, we do not wish to update the default game content, but to dynamically change

the educational material for other resources available on a remote location. Therefore, the default system will be kept intact for use cases where there is no personalization, or when there are technical issues that do not allow communication with the remote resources.

Unity3D

Unity 3D Real-time Development Platform [51] is used in many industries for creation of interactive environments, in 3D or 2D. It launched in 2005 as a game engine [52] and quickly became vastly popular with game developers. By 2018, almost half the games published on itch.io [53], a web gaming platform where independent developers publish games, was created with Unity, while 13.2% of games published on Steam [54], a digital distribution service with the largest market share, also used Unity3D as their engine [55]. Gradually, other industries started to utilize Unity as well, such as real estate development and even artists, [52] and Unity Technologies, the company behind Unity, welcomed and endorsed this, creating additional tools and renaming the game engine into real-time development platform. One of the biggest contributors to its popularity is the cross-platform capabilities it adopted since very early, supporting almost any platform current available, including cutting edge technologies. Additional contributors to its success are its beginner-friendly interface, the competence in professional needs, the effectiveness of the workflow and the ever-growing community, library of assets and library of tutorials that can cover any part of the engine or development process. All of the mentioned perks are provided for free, and additional premium features are available to professionals through subscription plans. For this endeavor we chose Unity3D due to all the aforementioned benefits, as well as the apprehension that, compared with other popular game engines, it is more likely to suit the needs of educational game developers.

NodeJS

NodeJS [56] is a runtime environment for Javascript. It is built utilizing the V8 Engine, an open source WebAssembly engine, published by google [57]. NodeJS is created as a way to write JavaScript for server-side programming. This locates NodeJS in a unique position, as it doesn’t only bring JavaScript, a language created, and traditionally used, for UI elements of dynamic web pages, but it also creates a single-threaded server, in contrast to most, if not all, other server-side languages, who follow a multi-threaded paradigm. It has become prominent in the web development circles, due to the benefits of using the same language for the server and

the client side, such as code reuse, the Node Package Manager, a built-in feature that allows developers to find and quickly import modules into their own application and it's simplicity for smaller projects. However, as expected, NodeJS is not a panacea, it is well suited for some scenarios, but for other there might be better alternatives. It is commonly preferred for application where incoming request are not CPU intensive and there are numerous concurrent connections to the server [58]. For our needs, NodeJS was selected for our RESTful API over other server-side solutions, as it provides simple integration with MongodDB, described below, and suits the needs of the application.

ReactJS

ReactJS [59] is a popular free and open-source JavaScript library, used for building interactive interfaces. Its architecture allows the production of single-page applications where data is dynamically updated through a virtual DOM. ReactJS is often compared to Vue.js and Angular, however the three are very different, as ReactJS is a library, Vue.js is a framework and Angular is a front-end platform [60]. They all utilize Javascript, but they shouldn't be compared as they are different in nature and goals. In our case ReactJS was chosen due to the smaller range of the pilot implementation. Not to be confused, ReactJS is a very good solution for massive web apps, but it is also very suitable for small projects, while Angular and Vue.js might be a little to much, as they are an entire platform and framework respectively.

MongoDB

MongoDB [61] is an open-source database system that uses JSON documents to store data, without the restriction of relational databases. Schemas are optional, and when they do exits it is up to the developer to decide how the relations will be expressed. It is very popular due to the fact that it stores data as JavaScript objects, readily available for web application written in JavaScript. Additionally, being NoSQL, it is a schemaless solution, providing flexibility that is impossible with relational databases. Naturally, these benefits do not come without drawbacks. The flexibility comes with an impact on consistency, and it is not the most optimized solution, especially when many aggregations are present [58]. MongoDB will generally be more suitable, compared to a relational database, when a flexible model is needed to store large amounts of denormalized data[62]. Therefore, it is selected for this implementation, as we need the

flexibility for the many kinds of content structure, for each game, that will result in a very large database, with numerous denormalized data.

Implemented Components

UnityWebRequest

Unity3D includes a toolbox for networking that offers techniques for connecting, downloading, uploading, etc. One of the classes in the toolbox is *UnityWebRequest*, which handles HTTP communication with web servers [63]. Communication employs the standard HTTP methods and *UnityWebRequests* utilizes additional classes, such as *DownloadHander* and *UploadHandler*, to send or receive data during the communication with web servers. Being a higher-level tool for HTTP communication, *UnityWebRequest* has methods and properties that arrange the appropriate communication format, stage HTTP headers etc. Yet, in our experience, communication with some web servers did not get established as expected. Various errors arose, and the functionality of *UnityWebRequest* did not allow for the proper customization. The most prevalent complication was the fact that the web server would only accept request in JSON format, while Unity will send XML format by default. This setting can be changed, through accessing the *UploadHandler* of the *UnityWebRequest*, and setting the *contentType* property. However, this introduces new complications and unpredictability, as there are cases where the setting will be overridden by defaults. Thus, to overcome this unpredictability, we created an extension class, presented in Figure 3.3.1. Asynchronous Web Request Class Diagram as *NileUnityWebRequest* and elaborated bellow, with the specific settings for the IOLAOS system [9], [64] with which we overcame the difficulties. Hence, creating an object with predefined communication format and headers, compatible with IOLAOS, with less parameters compared to the original class. Of course, this particular object will be compatible with any other web servers that communicate with the same format, namely JSON communication both in the request and the reply, but we have only tested it with IOLAOS. Additionally, through arguments, it can be utilized with other formats as well, but we have not tested this in practice either. As a result, it is easier for the developer to create a request, as they only need to instantiate the predefined object and provide the appropriate HTTP method, the service URL and any data that need to accompany the request.

Making UnityWebRequest Async

The Unity3D Manual suggests using coroutines when working with *UnityWebRequest*. Coroutines are a special kind of functions within Unity3D. They can only return one specific type, *IEnumerator* which is quite restrictive, but their benefit is that they can be used for calculations that need to span multiple frames, returning the control of program flow to where they have been called from, but continuing to execute themselves for as many frames as needed [63]. However, when it comes to asynchronous web requests, coroutines are not idea. The main problems that arise are (a) as coroutines can only return *IEnumerator*, any data fetched through the request cannot be returned. Instead, depending on the scope, global, private or reference variables are used, or all logic is included in the coroutine creating massive monolithic blocks of code (b) the architecture is compatible with game logic, but not web request, thus making the flow synchronous, while waiting for the response, (c) handling errors add additional complications, as we need to use a synchronous (try-catch) paradigm to inspect the asynchronous response. Web-based applications usually employ a different approach. All the above drawbacks can result in a game that has frozen while expecting a response from some server. Usually, such communications are rapid and users can not notice the delay, nonetheless network communications can be capricious, users might have a slow connection, the server might be overloaded, or a number of other complications might occur. Ergo, a different approach is needed.

In C# asynchronous programming for web requests is implemented using three keywords, *async*, *await* and *task*. This way a function, declared as *async*, pauses its own execution, when command identified by *await* is encountered, and releases its thread, allowing the execution of the program to continue, while waiting on specific events to occur. An incomplete task is returned to the caller of the method, who continues execution of the program. The async method will continue executing when the expected event has occurred (in this case, the response from the server has been received) [65]. By default, Unity3D (version 2019.4.17.f1) uses an older version of .NET, namely .NET Standard 2.0, were *async/await/task* was not yet included, but this can be modified, and .NET 4 can be used instead, allowing these keywords to be used within Unity scripts. Still, *UnityWebRequest*, and its methods, are not created with async/await in mind and are thus not compatible, i.e. we can't use *SendWebRequest* method with *await*, as there is no *GetAwaiter*() to call. So, we created another extension, to make the *SendWebRequest* compatible

with *await*. This way, we have an asynchronous call to the web server, execution of the game continues while we wait for the response to be fetched, and when the response is received, it is returned to the central thread of the game logic to be checked for errors, evaluated and acted upon. The additions implemented are illustrated in the class diagram of **Error! Reference s ource not found.**. *NileUnityWebRequest*, the class described in the previous subsection, inherits *UnityWebRequest* and creates a web request with particular formatting. By default, *UnityWebRequest* will utilize a *UnityWebRequestAsyncOperation*, but due to the flow of coroutines described above, the program flow will not be asynchronous. To solve this, we have to both call the *SendStatement* method of *UnityWebRequest* as an awaitable task, but we also have to make *SendStatement* and awaitable task to be able to do this. Therefore, we have created the *AsyncOperationAwaiter* class, which provides the *GetAwaiter* method that is the requisite for an operation to be asynchronous. Finally, *WebRequestProgressNotifier* and *ProgressUpdater* are additional classes that provide the necessary insight into the state of the asynchronous operation and notify the main flow of execution when the task is completed. As a result, we have achieved to both simplify the procedure of writing web request, and made them function in a truly asynchronous manner, feeing up the main thread for the rest of the game to continue execution.

Trilib Library

Unity3D has four mechanisms for importing assets, such as textures, audio and prefabs, at runtime. Namely, the Resources Folder, the Streaming Assets folder, the Asset Bundles and the Addressable Asset System. Each has their benefits and drawbacks and deciding which approach to utilize depends on the unique circumstances of each project and problem to be solved. However, when it comes to 3D models, all the aforementioned techniques have one common major drawback. Though it is possible to import 3D models through these tools, all methods require the model to be imported in a specific way as an Asset, to then be utilized by said methods. In other words, importing 3D models with any of these techniques requires a developer to prepare each and every 3D model as an asset, in a Unity proprietary format, before it can be used as a dynamic resource. In our case, we wish to detach the procedure from the game developer, therefore 3D models found on the web or created by anyone interested, should be able to be utilized, without the extra workflows and technicalities implied by the need for the creation of a unity asset before it can be utilized by educators and other stakeholders. Thus, TriLib was

selected as a tool that would help us in creating a dynamic loading system for 3D models from the world wide web.

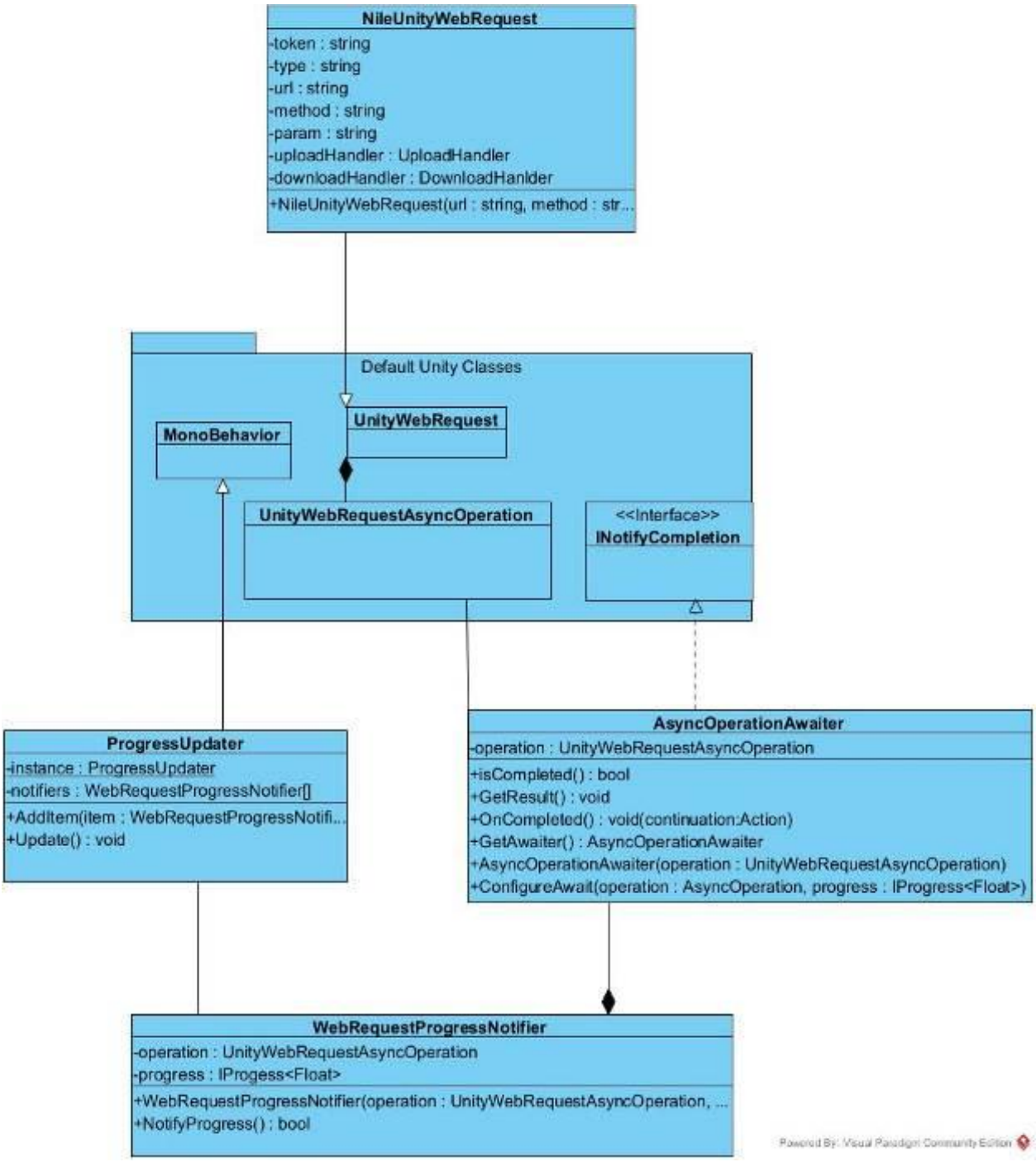

Figure 3.3.1. Asynchronous Web Request Class Diagram

Trilib [66] is a package for Unity3D real-time engine, available through the Unity Asset Store. Using TriLib allows us to import 3D models at runtime, which is not possible with the built-in capabilities of the engine. TriLib supports cross-platform development, including the major desktop and mobile operating systems, a wide selection of 3D formats, such as FBX, OBJ, 3MF and others. Additionally, zip files that include the 3D model along with the textures and animations are supported, which is very helpful for the procedure of uploading and managing complex assets.

Even though TriLib is a plug-and play library, effectively employing it calls for some additional infrastructure to accommodate the flow of events and handle the amount of models that need to be loaded from remote locations and the resulting assets. Figure 3.4.2 - Models Import Subsystem Class Diagram illustrates the class diagram of the Model Loader subsystem we developed. Following the guidelines of Trilib documentation and sample code we created the *ModelImporter* class, that has all the methods need to load models from remote resources. Additionally, we included attributes and methods to instantiate models in the game scene. As part of the procedure to instantiate GameObject, their tag is set to a value, "TrilibImported", so that they can be identified through usual practices. In most cases, references to these GameObjects will be saved, and they might be modified in many ways, including the tag, but we felt that this was a necessary detail for generalizing the solution and providing safety nets. Web requests for fetching the models are done through the *NileUnityWebRequest*, described above. Finally, *ScenePersonalizer* is the class that knows what objects can be exchanged for others, and where they should be placed in the scene. These objects are represented by a *PersonalizebleModel* class. The *ScenePersonalizer* is the mediator between the core of the game and this subsystem and can either be used as a default procedure to load all models, or be expecting specific requests, depending on what would be best for each game and scene.

Rest API & Database

This part of the system was created in collaboration with other members of NILE Lab [67], in an attempt to keep a generalized path that would serve this project as well as other, future, endeavors. As mentioned above, the server-side application is developed with NodeJS and MongoDB is both the database and the file server from the architecture described in Figure 3.1 - Generalized Architecture. The main goal of this module is to receive requests from the

games, utilize the data provided with the request to locate the corresponding material in the database and reply with the appropriate format and content. The data provided will be a learning session ID, which will be bound to a specific game and personalized educational material. Thus, the API is game-agnostic and handles the content as part of a learning session. For the moment, the learning session is expressed as a unique Token, which servers the goal of cross-utilization between this platform and any other potential usage of the API. Even though it might seem simpler to request content based on the game, the learning session is more appropriate, as each learning session will only have one game and specific personalized content, while a game might have a multitude of personalized content from various learning sessions. Thus, if this approach was followed, there would be a need for additional levels of filtering and querying, resulting in longer wait times and wasted CPU cycles. Admittedly, the API is quite small for the moment, as the pilot implementation calls for a small scale. This API will grow when the editor is further developed, resulting in needs for multiple services to write material to the database, retrieve data with sophisticated filters, utilize authentication, etc.

For our pilot implementation, and the requirements set forth by the utilization of ThimelEdu, the data that are stored in the database includes informational texts, informational pictures and a 3D model. Specifically, for each informational text we record an id, the associated interest point, title, the text itself, the school type for which the text is appropriate and the language in which it is expressed. Additionally, for each image, we hold the image itself and some metadata, specifically, associated interest point, title and appropriate school type. Finally, for the 3D model we only hold the model itself, as there is no use for any metadata within the game. However, this structure will depend on any game that will utilize the framework in the future, creating different context.

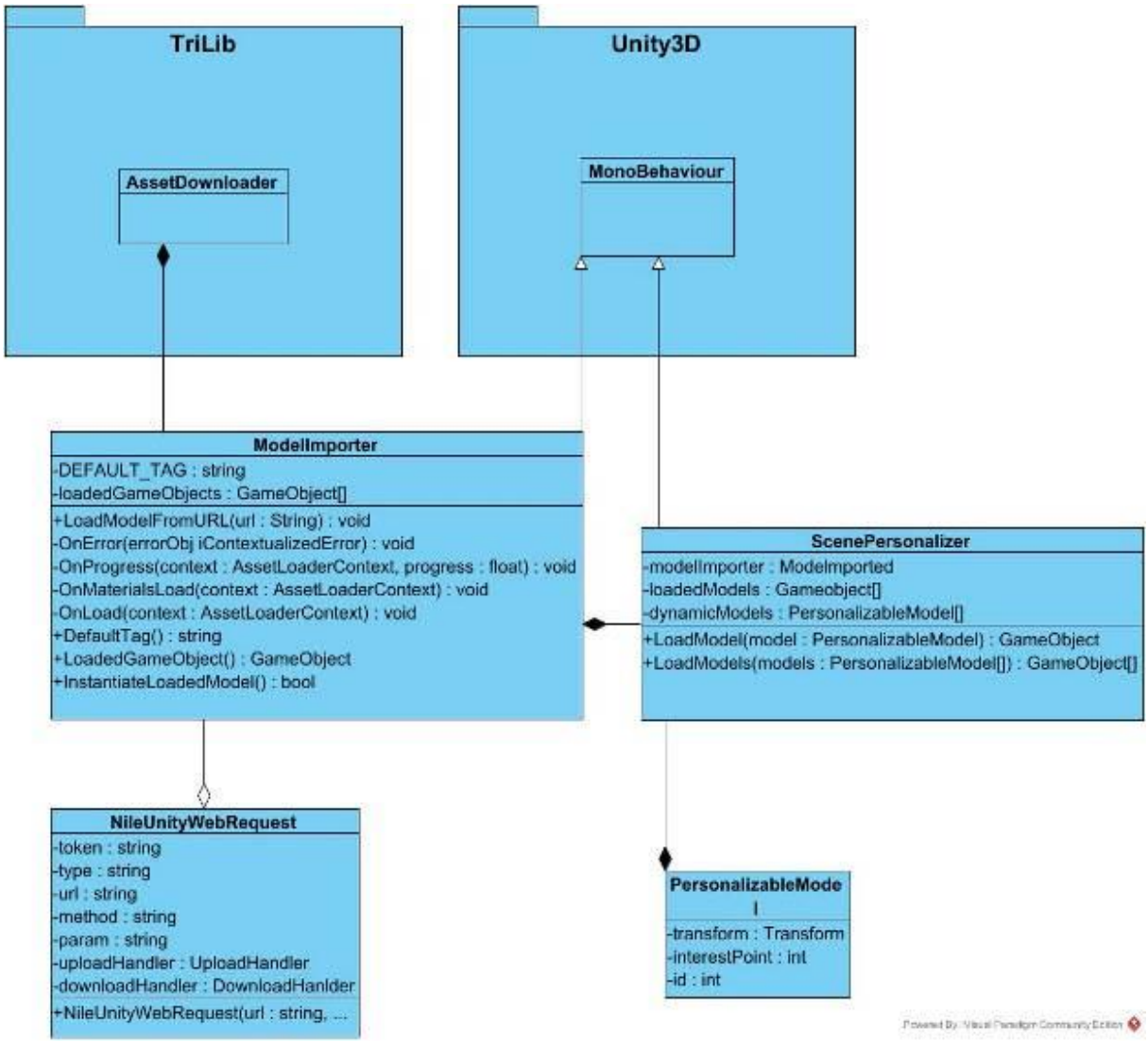

Figure 3.4.2 - Models Import Subsystem Class Diagram

ReactJS Front-End

The front end refers to the web site where educators can create and edit content for personalized playing sessions. This part of the system was also developed as a joint effort with other members of NILE Lab [67], under the instructions of the author. Like the Rest API and Database, this also serves as a double project, offering game hosting and customization tools. For this thesis, only the customization tools are relevant and will be examined. This website is accessible by educators and game developers.

Game developers can add a game to the system. To do so, they will have to provide a name, URL where the game can be found by interested educators, a poster to be showcased, and

a description of the game. Additionally, they can select a category (serious game, puzzle, mind games are available for the time being). Lastly, they must express their game content structure in JSON format, following the guidelines provided (see JSON Format Guidelines). This will dynamically create a webpage where content can be created and edited, following the structure provided.

Educators can select a game they are interested in, see details about it and if they wish to do so, they will be able to create a custom game session. The page dynamically created in accordance to the content structure will be presented to the educator, who can add a Title to the game session and will be presented with any buttons and input fields needed to create their custom game experience.

Types available for the content structure include, but are not limited to, strings, numbers, arrays, enums, colors, choice (radio buttons, checkboxes) objects and images. Extensions must be done to properly support 3D models, audio, video and other media types, but even with the current available data types every need can be satisfied, through proper utilization of arrays and objects. Adding the support would make it more apparent to educators and is thus a user experience upgrade, but not a technical necessity.

Chapter 4 - Pilot Use Case

The pilot implementation described above has confirmed the potential of the proposed system and provides good ground for future developments and expansions. To further prove the strengths of the proposed system and explore any overlooked issues, we created a representative scenario of use with ThimelEdu, the game modified. Unfortunately, working on a game that did not include this extensive level of personalization in its initial design is not ideal to showcase the possibilities that our proposed framework offer. Nonetheless, even at the restricted degree possible with this game, the technical prospects are clearly distinguishable. Future serious games, designed to employ the framework and create the best possible result, will be much better fitted and show new grades of dynamic content that might be unthinkable to the author at the moment of writing.

Representative scenario

To showcase the results of the implemented portion of our proposed framework, we considered a specific scenario through which we could observe the differences in game. As we expect from the scenario described in Exhibit 1, our player, Maria, should be presented with different educational content throughout two distinct educational sessions.

Exhibit 1: Learner Maria is a student in the 4th grade. Her teacher in history, John, decided to use ThimelEdu as an assisting tool for teaching the topic of ancient Greek theatre. By playing the game, Maria and her classmates get familiar with the architecture of the ancient Greek theater, and the tools used during plays, through the educational material provided with the game. The next day, John asks his pupils to play the game again, only this time he has used IOLAOS and our framework to create a learning session and create personalized content for his pupils. Maria and her classmates join the game again, only to find different information.

Content Customization

Firstly, we will examine the procedure John, the Teacher followed to customize the content for the second learning session. For this, we assume he is already familiar with the game and has an IOLAOS account. His next step is to visit the front end webpage of our platform and login using his IOLAOS account. He is then presented with the landing page shown in Figure 4.1 - Landing Page. In this case, the game John is interested in is promoted on the landing page. In case the game he was interested in was not located on this page, he could select the game category from the left lateral bar and browse a more extensive list of available games.

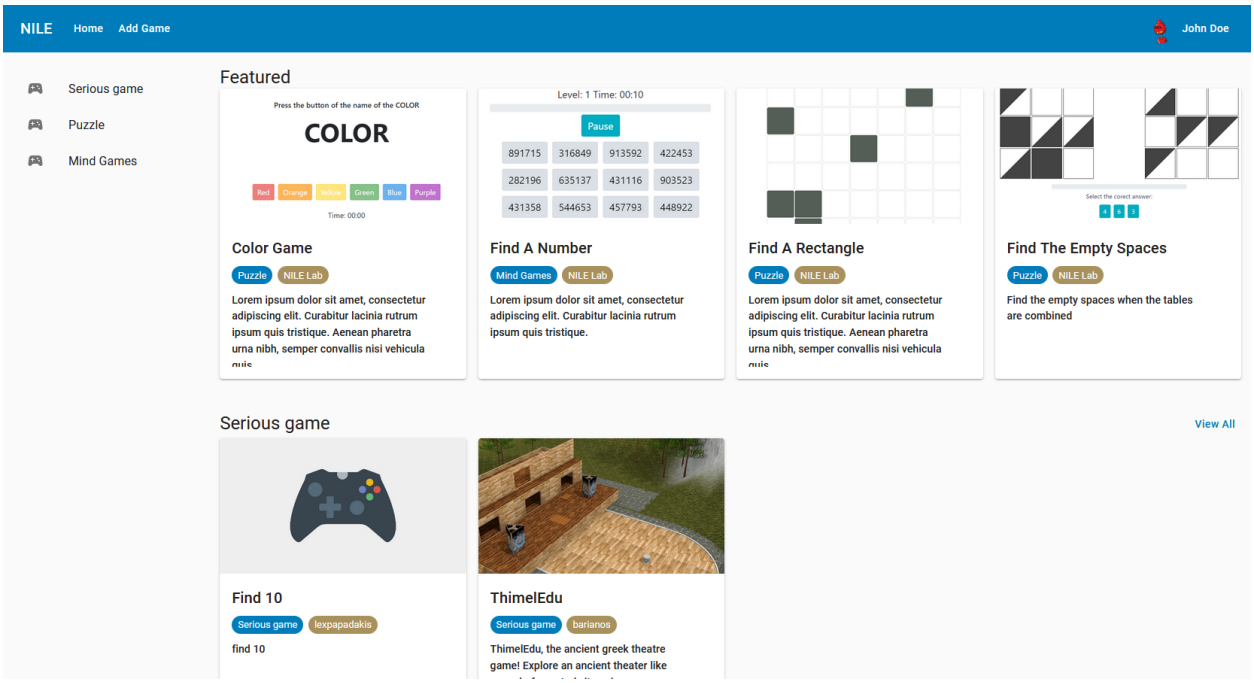

Figure 4.1 - Landing Page

John selects ThimelEdu and is presented with the modal shown in Figure 4.2 - Game Options Modal. The available option in this is to Play the game, which would forward John to the link related to the game, to create a custom game or to give a token and play a customized version of the game. John wants to create a custom experience for his students, so he chooses to customize the game. From there he is presented with the available options that are created from the JSON provided by the game developer, specifically “Add Informative Text”, “Add Informative Picture” and “Add 3D Object of Interest”, as shown in Figure 4.3 - Customization Options. Additionally, John must select a game title for his personalized gameplay.

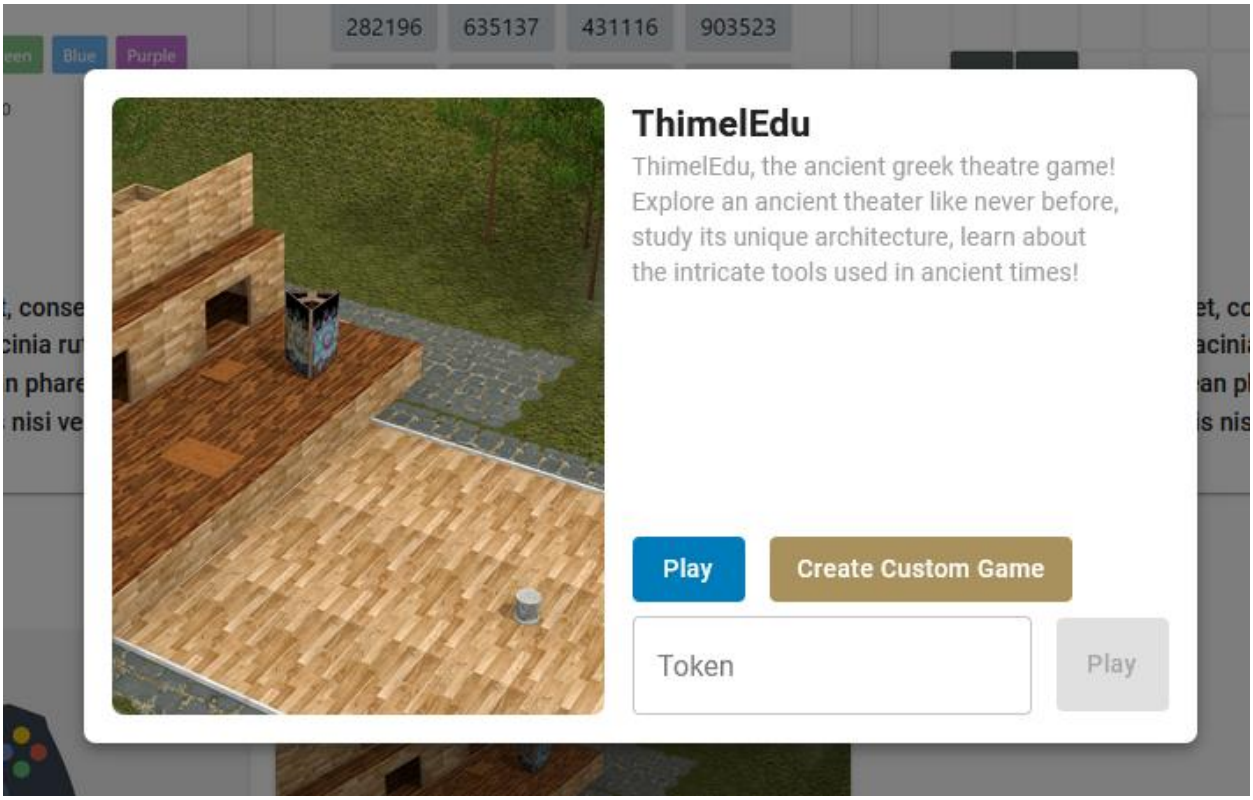

Figure 4.2 - Game Options Modal

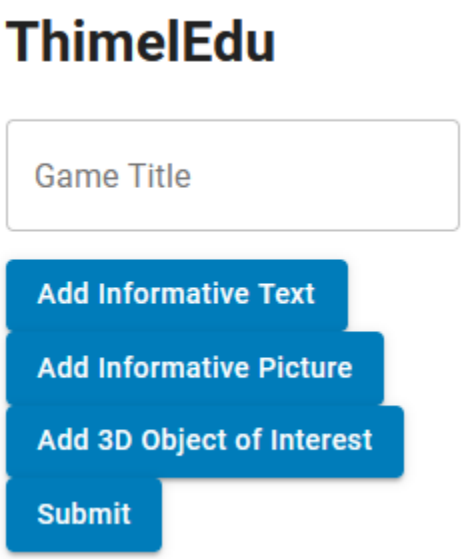

Figure 4.3 - Customization Options

By extending each selection he is presented with the exact information and media required for each of the selection. With this in mind, he fills all the inputs with the information he wants his pupils to study, as shown in Figure 4.4 - Providing information and media. In this specific screenshot we can discern that he has decided to teach his pupils about the sandals. How this affects the game and the learning process will be better showcased bellow, but here we can see that he has decided to create an informational text, add images and even a 3D object of a sandal, providing the essential media but also metadata, like a title (“Sandals”), targeted school type, language of text, etc.

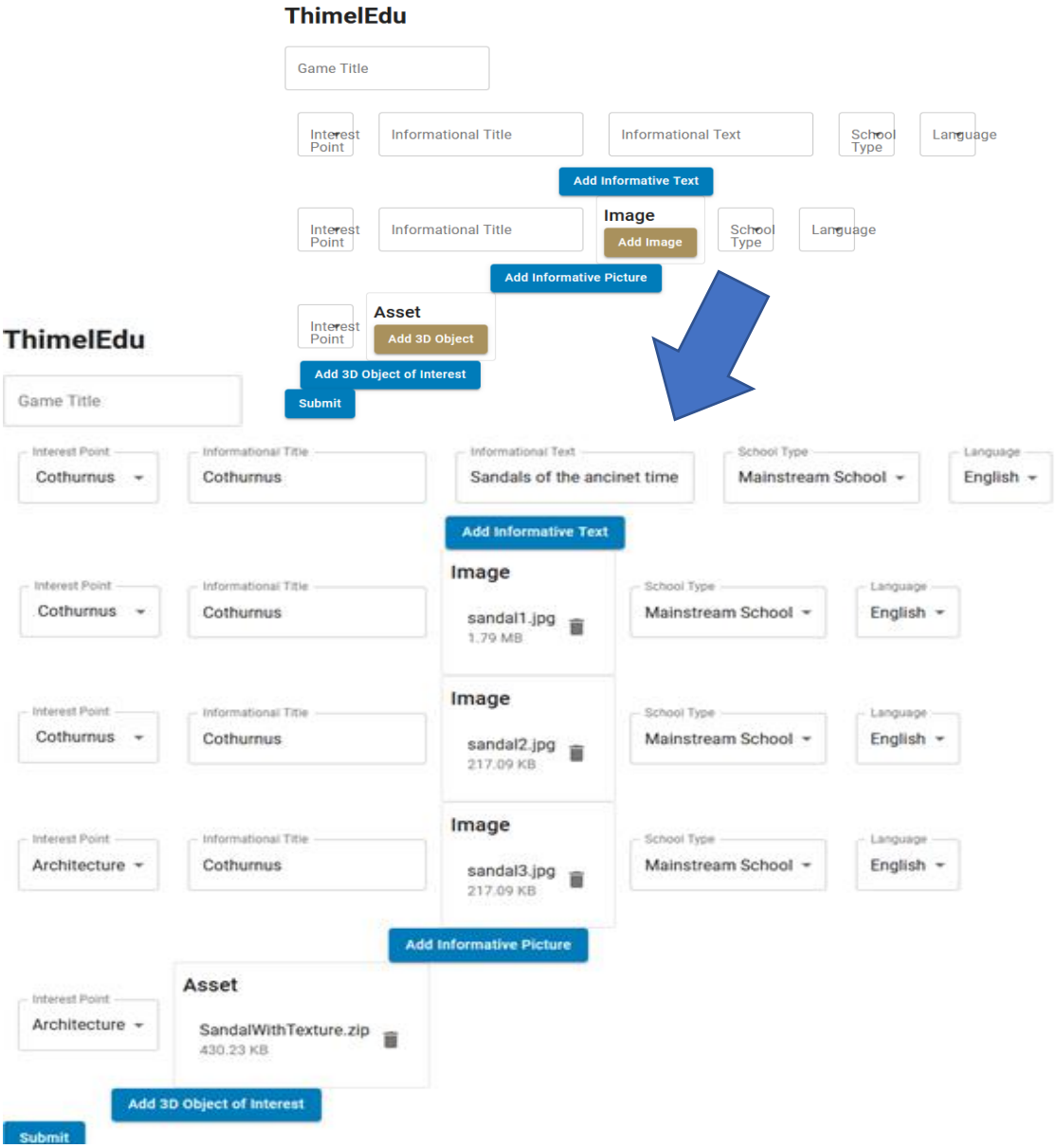

Figure 4.4 - Providing information and media

He continues to add all other educational elements he wants to add to his session. When he is finished, he pushes submit and is present with a Token, shown in Figure 4.5 - Game Created Message, that will enable access to the game for his pupils. So, a few days later, when it's time for his students to play the customized version of the game, he will share this token and his students will be presented with a different version of the, already familiar, game.

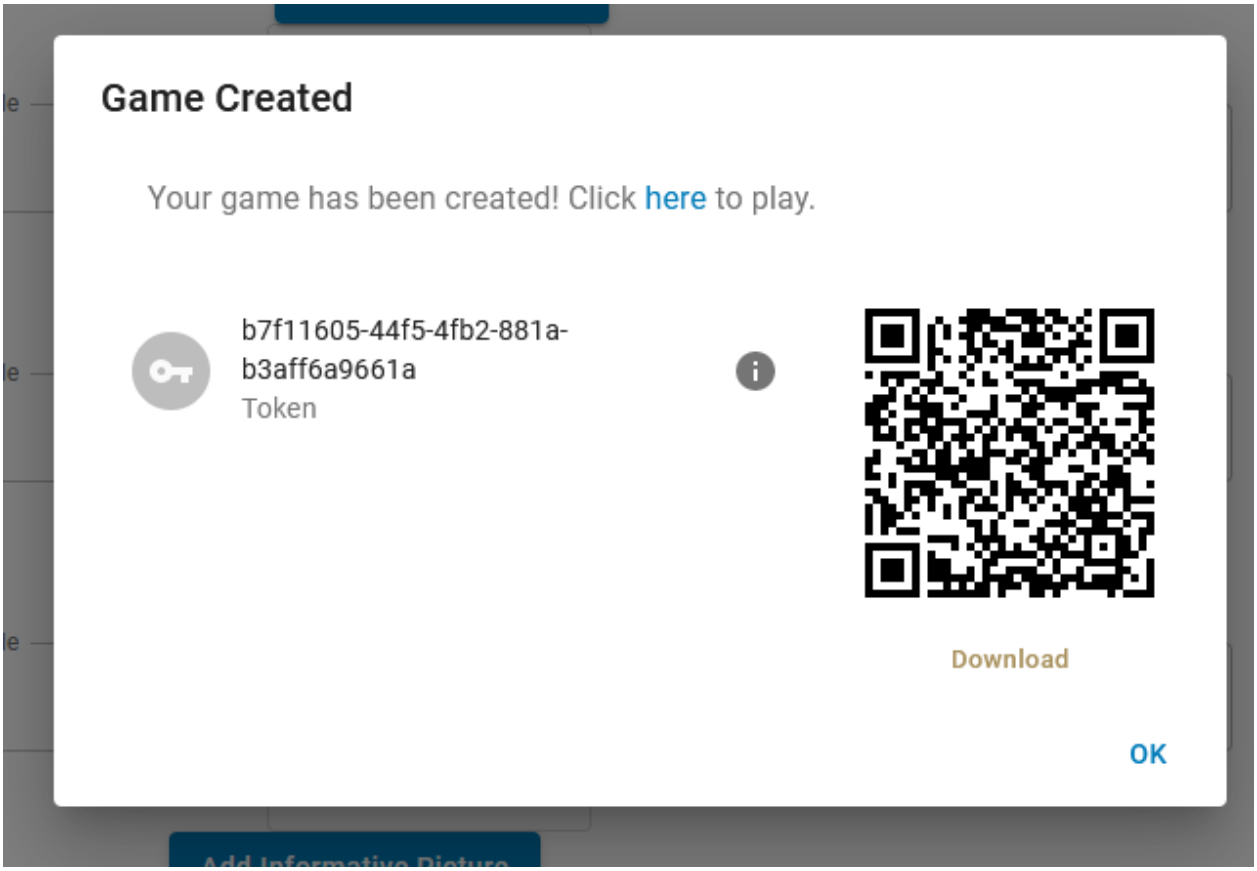

Figure 4.5 - Game Created Message

Gameplay Differentiation

As defined by our scenario, Johns class will play ThimelEdu in two learning sessions. During the first they will experience the game as it was created and shipped by the developers, while the second time they will play the modified version that their teacher has created. Among other adjustments, John has decided to generalize the topic of cothurnus, a shoe used during ancient tragedy plays, into the sandals, of which cothurnus are a specific kind. Above we examined the procedure John followed to add the sandals into the game, now let's see the differences that Maria, one of his students, will experience.

As shown in Figure 4.6 - Cothurnus and Sandal, the default in game artefact for user interaction is a cothurnus, which is replaced by a sandal that John provided. Let us note that functionality doesn't change, i.e., interacting with the object will trigger related information to appear, just as it would with any other default or modified object.

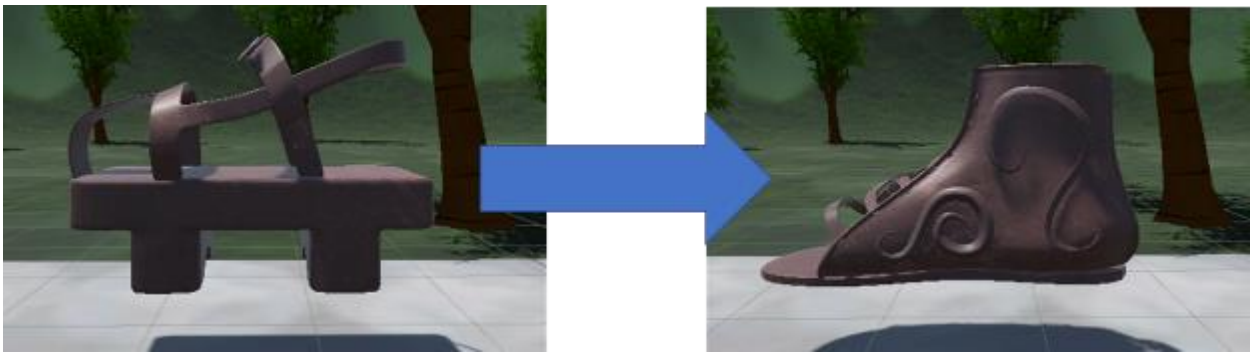

Figure 4.6 - Cothurnus and Sandal

Following the scenario, and what Maria experiences each day, at first the cothurnus is the artefact that she will discover in the game, as dictated by the default content that the game developers have authored the game with. In Figure 4.7 - Cothurnus in-game interaction, the flow of events is illustrated, first Maria locates an artefact, the cothurnus, and when interacting with the object an informational screen appears. In this case we can distinguish 3 drawings of cothurnus, and a text that reads "*The Cothurnus were shoes used by tragic actors. They were Tall, had soles that made the actors look taller and were usually built from wood*.". From this, we can deduct, that during the first day the class learns about the cothurnus, their structural details, that they were used by tragic actors, and why. The next day, Maria once again plays the game, but this time within a IOLAOS learning session, with educational material provided by her teacher. Today the cothurnus object is replaced by a different, yet similar, type of shoe, a sandal. This draws Marias attention, and the same chain of actions as above is followed, and illustrated in Figure 4.8 - Sandal in-game interaction. Upon interaction with the new artifact that is located in the same place as the cothurnus was yesterday, Maria is now presented with a different set of information. The title of the informational screen is the same as yesterday, Cothurnus, but the pictures presented are different and so is the provided text. There are two pictures of actual sandals, and one of a sandal on an ancient sculpture. The new text reads "*Sandals of the ancient time were distinguished into two kinds, baxea and cothurnus. The first was simpler sandals,*

made from willow leaves and twigs, usually worn in comedy plays. The later was more like a boot, with thick soles to add to the stature and was usually worn in tragedy plays.".

Through this second learning session, Maria and her classmates has learned a little more about ancient footwear, about the sandals in general, and how there is a difference between the shoes used in comedy to those used in tragedy.

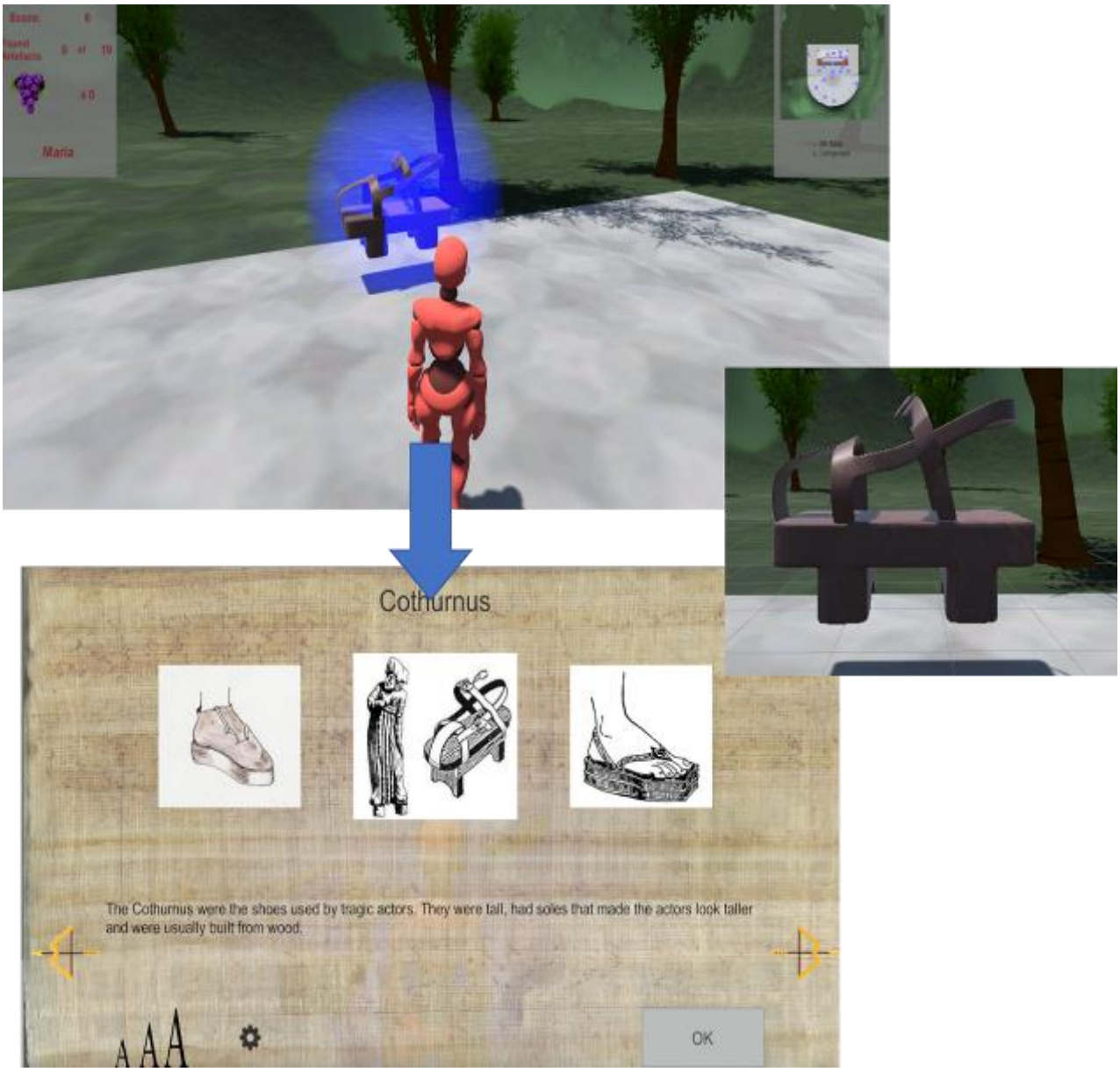

Figure 4.7 - Cothurnus in-game interaction

To further illustrate how the framework works we will examine another modification made by the teacher. **In Figure 4.9 - Thimele Default Content** we can see a snapshot from

Marias interaction with a second artefact during the first day of gameplay. Maria has found the Thimele, and altar to the god Dionysus, and interacted with it, thus being presented with a selection of educational texts and pictures about the Thimele. As we can see, she is presented with three pictures and only one text of a simple couple of sentences. The text mentions "*At the center of the orchestra was the altar of Dionysus, the Thimele. Behind it stood the piper and the prompter*" However, we can also see arrow buttons (bow with arrow icon) to the left and the right of both elements. This signifies that there is additional content that the player can scroll through to get additional information. Maria continues to explore the game and finds the other artefacts, always being presented with the default content prepared by the game developers.

Figure 4.8 - Sandal in-game interaction

The next day, Maria once again plays the game, where cothurnus is not the only updated artefact. In Figure 4.10 - Thimele Teacher Defined Content we can observe another difference Maria is presented with, compared to the previous day. She is presented with 3 pictures, of which only one was part of the previous days' selection. The teacher apparently wanted to reuse this particular picture for some educational reason and decided to include it in her selection. He could just as well have used only new pictures, or only the pictures provided by the game, or any amount of new and default pictures. The same principle applies of course to any other type of content that teachers wish to personalize. Additionally, we can see that the text Maria is presented with is different from last days, as the teacher decided to only add his own texts for this learning session. The text teaches a detail not mentioned in the previous session material, specifically "*Thimeles history goes as far back as the 7th century BC and was always an altar to Dionysus. Its shape could vary but was usually either square or cylindrical*". Lastly, we notice that the arrow buttons are not present anymore. This happened because there are only 3 pictures and one text available for the thimele artefact. In other words, for this specific case, the educator wished to stress one particular point, that the thimele could have a box or a cylindrical shape, and thus only wrote one text emphasizing this information, and provided a picture her pupils would be familiar with from the previous learning session, and two new pictures illustrating the alternative shape. It should be clarified that the buttons for scrolling through pictures and texts are automatically enabled or disables from the game, depending on the number or materials available for each artefact, it has not been modified by the educator. Additionally, one element that should be dynamic, but as we saw in Figure 4.8 - Sandal in-game interaction is the title of the informational screen. This was a detailed overlooked when modifying the game to be dynamic. From the above we safely state that our Unity3D adaptation system works as expected and the framework provides the grounds for adaptable learning sessions in serious games.

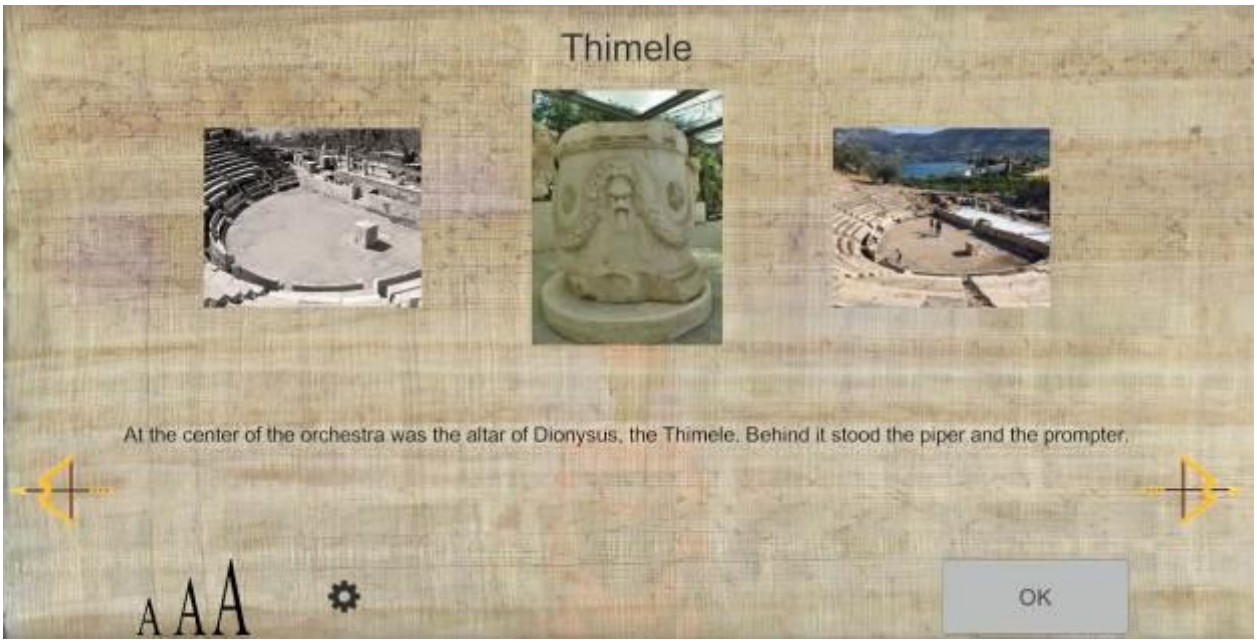

Figure 4.9 - Thimele Default Content

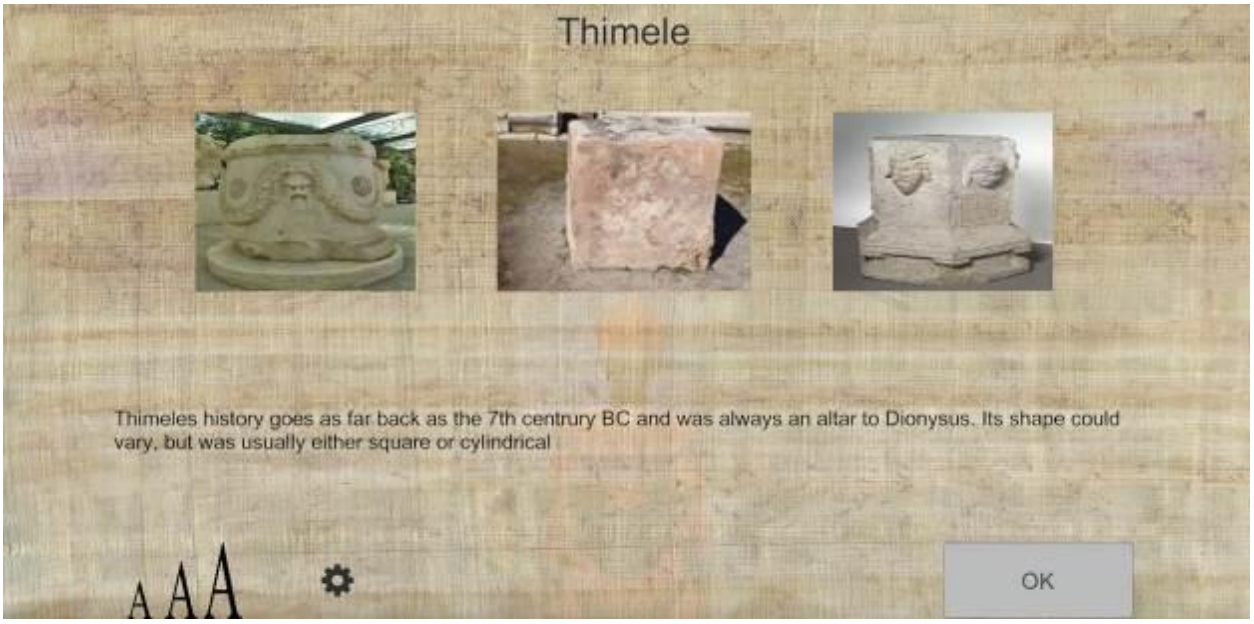

Figure 4.10 - Thimele Teacher Defined Content

Chapter 5 - Conclusion and Future Work

Education is rapidly evolving and catching up with the booming advances in technology. Game-based learning is all the more present in official and unofficial learning settings and serious games are gaining ground daily. Nonetheless, serious games have yet to be defined in a conceptual manner. Research and conversations are conducted at large, postulating and testing ideas that drive the field forwards. Through all this research we are usually hunting noble goals that are just beyond reach, often bringing them slightly closer to reality. However, in this procedure, we often neglect goals much more attainable that have immediate impact. Game development, and especially educational game development, depends on a wide collaboration of professionals and fields. Additionally, designing a game adaptive enough to be effective for every learner, with any background and under all circumstances is a challenging goal. Through this thesis we located a shortcoming of the serious games industry in utilizing current technologies to create adaptable games at runtime and through this insight we created an abstract idea on which we build the requirements and design of an innovative project that addresses the aforementioned issue. Additionally, we developed a pilot implementation, partially fulfilling the idea into a tangible project that can serve as an initial foundation for future expansions that will realize the entire framework. We hope that the proposed framework will elevate the learning procedure and the benefits for students, while also greatly assisting collaboration of educators and developers. Additionally, we expect this framework to serve as a guideline for content design and game development that will reduce time and finances invested in recurring tasks and thus facilitate the creation of better games offering more interesting experiences.

In future work, our priority is to extend the Content Editor and REST API, realizing the entire framework. Additionally, we wish to create the necessary infrastructure for more popular game engines and for web-based serious games that don’t utilize a game engine. Additionally, we wish to improve the current implementation, by refactoring the procedures that load the remote object into Async procedures, moving the entire solution into an asynchronous paradigm, and further optimizing the architecture. Lastly, we aim to utilize semantic notation with the data handled by the framework, so that there can be further future developments in automating and tracking.

References

[1] C. C. Abt, “Serious Games.” 1970, doi: 10.1109/VS-GAMES.2009.8.

[2] M. Ulicsak, “Games in Education: Serious Games,” *A Futur. Lit. Rev.*, p. 139, 2010, [Online]. Available: http://www.futurelab.org.uk/projects/games-in-education.

[3] M. Zyda, “From visual simulation to virtual reality to games,” *Computer (Long. Beach. Calif).*, vol. 38, no. 9, pp. 25–32, 2005, doi: 10.1109/MC.2005.297.

[4] S. Arnab *et al.*, “Mapping learning and game mechanics for serious games analysis,” *Br. J. Educ. Technol.*, vol. 46, no. 2, pp. 391–411, 2015, doi: 10.1111/bjet.12113.

[5] T. M. Connolly, E. A. Boyle, E. Macarthur, T. Hainey, and J. M. Boyle, “A systematic literature review of empirical evidence on computer games and serious games,” *Comput. Educ.*, vol. 59, no. 2, pp. 661–686, 2012, doi: 10.1016/j.compedu.2012.03.004.

[6] F. Bellotti, B. Kapralos, K. Lee, P. Moreno-Ger, and R. Berta, “Assessment in and of serious games: An overview,” *Adv. Human-Computer Interact.*, vol. 2013, 2013, doi: 10.1155/2013/136864.

[7] J. Breuer and G. Bente, “Why so serious? On the relation of serious games and learning,” *Eludamos. J. Comput. Game Cult.*, vol. 4, no. 1, pp. 7–24, 2010.

[8] R. Daconceicao, C. Locke, K. Cooper, and C. S. Longstreet, “Semi-automated serious educational game generation: A component-based game engineering approach,” *Proc. CGAMES 2013 USA - 18th Int. Conf. Comput. Games AI, Animat. Mobile, Interact. Multimedia, Educ. Serious Games*, pp. 222–227, 2013, doi: 10.1109/CGames.2013.6632637.

[9] N. Vidakis and S. Charitakis, “Designing the Learning Process,” in *Proceedings of the 10th International Conference on Subject-Oriented Business Process Management - S-BPM One ’18*, 2018, pp. 1–11, doi: 10.1145/3178248.3178254.

[10] L. Hanes and R. Stone, “A model of heritage content to support the design and analysis of video games for history education,” *J. Comput. Educ.*, vol. 2, 2018, doi: 10.1007/s40692-018-0120-2.

[11] C. Malliarakis, F. Tomos, O. Shabalina, and P. Mozelius, “Andragogy and

E.M.O.T.I.O.N.: 7 Key Factors of Successful Serious Games," in *European Conference on Games Based Learning*, 2018, no. October, p. pp.371-378.

[12] S. De Freitas and M. Oliver, "How can exploratory learning with games and simulations within the curriculum be most effectively evaluated?," *Comput. Educ.*, vol. 46, no. 3, pp. 249–264, 2006, doi: 10.1016/j.compedu.2005.11.007.

[13] N. Suttie *et al.*, "In persuit of a 'serious games mechanics' : A theoretical framework to analyse relationships between 'game' and 'pedagogical aspects' of serious games," *Procedia Comput. Sci.*, vol. 15, pp. 314–315, 2012, doi: 10.1016/j.procs.2012.10.091.

[14] Rustici Software, "Experience API." https://xapi.com/overview/ (accessed Jan. 26, 2018).

[15] M. Paulsen, "Online Education Systems : Discussion and Definition of Terms," *NKI Distance Educ.*, pp. 1–8, 2002, [Online]. Available: https://www.edutubebd.com/file_resource/1368197236online education system.pdf.

[16] B. Huynh-kim-bang, J. Wisdom, and J. Labat, "Design Patterns in Serious Games : A Blue Print for Combining Fun and Learning Introduction : Making Learning Fun," *J. Comput. Game Cult.*, pp. 1–18, 2010, doi: 10.1080/0142569880090306.

[17] J. Stewart *et al.*, *The Potential of Digital Games for Empowerment and Social Inclusion of Groups at Risk of Social and Economic Exclusion : Evidence and Opportunity for Policy*. 2013.

[18] W. van der Vegt, W. Westera, E. Nyamsuren, A. Georgiev, and I. M. Ortiz, "RAGE Architecture for Reusable Serious Gaming Technology Components," *Int. J. Comput. Games Technol.*, vol. 2016, pp. 1–10, 2016, doi: 10.1155/2016/5680526.

[19] C. Hurtado, G. Licea, and M. Garcia-Valdez, "Integrating Learning Styles in an Adaptive Hypermedia System with Adaptive Resources," *Stud. Syst. Decis. Control*, vol. 143, pp. 49–67, 2018, doi: 10.1007/978-3-319-74060-7_3.

[20] P. Brusilovsky, "Adaptive hypermedia for education and training," *Adapt. Technol. Train. Educ.*, pp. 46–66, 2012, doi: 10.1017/CBO9781139049580.006.

[21] M. Alshammari, R. Anane, and R. J. Hendle, "An E-learning investigation into learning style adaptivity," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2015-March, pp. 11–20, 2015, doi: 10.1109/HICSS.2015.13.

[22] C. Troussas, A. Krouska, and C. Sgouropoulou, "Collaboration and fuzzy-modeled personalization for mobile game-based learning in higher education," *Comput. Educ.*, vol.

144, p. 103698, Jan. 2020, doi: 10.1016/j.compedu.2019.103698.

[23] N. Vidakis, A. Barianos, G. Xanthopoulos, and A. Stamatakis, “Cultural Inheritance Education Environment: The Ancient Theater Game ThimelEdu,” in *12th European Conference on Games Based Learning, ECGBL 2018*, 2018, pp. 730–740.

[24] S. Papadakis, M. Kalogiannakis, and N. Zaranis, “Educational apps from the Android Google Play for Greek preschoolers: A systematic review,” *Comput. Educ.*, vol. 116, pp. 139–160, Jan. 2018, doi: 10.1016/j.compedu.2017.09.007.

[25] W. B. Frakes and K. Kang, “Software reuse research: Status and future,” *IEEE Trans. Softw. Eng.*, vol. 31, no. 7, pp. 529–536, 2005, doi: 10.1109/TSE.2005.85.

[26] W. Frakes and C. Terry, “Software reuse: metrics and models,” *ACM Comput. Surv.*, vol. 28, no. 2, pp. 415–435, 1996, doi: 10.1145/234528.234531.

[27] W. van der Vegt, E. Nyamsuren, and W. Westera, “RAGE Reusable Game Software Components and Their Integration into Serious Game Engines,” 2016, pp. 165–180.

[28] GALA, “Roadmap on Serious Games,” pp. 1–33, 2014.

[29] S. Downes, “Learning Objects : Resources For Distance Educa- tion Worldwide The Need for and Nature of Learning Objects,” *Int. Rev. Res. Open Distance Learn.*, vol. 2, no. 1, pp. 1–35, 2001, doi: 10.19173/irrodl.v2i1.32.

[30] RAGE project, “RAGE project,” [Online]. Available: http://rageproject.eu/.

[31] Serious Game Society, “Serious Game Society.” https://seriousgamessociety.org/about/ (accessed Jan. 25, 2019).

[32] F. M. Dagnino, M. Ott, F. Pozzi, and E. Yilmaz, “Serious Games Design: Reflections from an Experience in the Field of Intangible Heritage Education,” *11th Int. Sci. Conf. eLearning Softw. Educ.*, vol. 2015, pp. 57–64, 2015, doi: 10.12753/2066-026X-13-131.

[33] Serious Game Society, “GLEANER.” https://e-ucm.github.io/gleaner/ (accessed Jan. 27, 2018).

[34] Serious Game Society, “SGREF.” http://www.sgref.com/ (accessed Jan. 27, 2019).

[35] B. Akkoyunlu and M. Y. Soylu, “A study of student’s perceptions in a blended learning environment based on different learning styles,” *Educ. Technol. Soc.*, vol. 11, no. 1, pp. 183–193, 2008, doi: 10.1007/s00217-010-1351-2.

[36] C. Mulwa, S. Lawless, M. Sharp, I. Arnedillo-Sanchez, and V. Wade, “Adaptive educational hypermedia systems in technology enhanced learning,” *Proc. 2010 ACM*

Conf. Inf. Technol. Educ. - SIGITE '10, p. 73, 2010, doi: 10.1145/1867651.1867672.

[37] M. Ainscow, "Developing inclusive education systems: what are the levers for change?," *J. Educ. Chang.*, vol. 6, no. 2, pp. 109–124, Jun. 2005, doi: 10.1007/s10833-005-1298-4.

[38] N. Peirce, O. Conlan, and V. Wade, "Adaptive educational games: Providing non-invasive personalised learning experiences," *Proc. - 2nd IEEE Int. Conf. Digit. Game Intell. Toy Enhanc. Learn. Digit. 2008*, pp. 28–35, 2008, doi: 10.1109/DIGITEL.2008.30.

[39] "IOLAOS." https://seriousgame.teicrete.gr/ (accessed Jan. 25, 2019).

[40] C. Karagiannidis, D. Sampson, and F. Cardinali, "Integrating adaptive educational content into different courses and curricula," *Educ. Technol. Soc.*, vol. 4, no. 3, pp. 37–44, 2001.

[41] J. Doboš and A. Steed, "3D revision control framework," in *Proceedings of the 17th International Conference on 3D Web Technology - Web3D '12*, 2012, p. 121, doi: 10.1145/2338714.2338736.

[42] J. Doboš, "3D Repo: Version Controlled Repository," p. 6, [Online]. Available: http://3drepo.org/wp-content/uploads/2015/03/3drepo-poster-portrait-aag.pdf.

[43] J. Behr, P. Eschler, Y. Jung, and M. Zöllner, "X3DOM," in *Proceedings of the 14th International Conference on 3D Web Technology - Web3D '09*, 2009, p. 127, doi: 10.1145/1559764.1559784.

[44] 3D Repo, "3Drepo.io." https://www.3drepo.io (accessed May 06, 2021).

[45] S. Friston, C. Fan, J. Doboš, T. Scully, and A. Steed, "3DRepo4Unity," in *Proceedings of the 22nd International Conference on 3D Web Technology*, Jun. 2017, pp. 1–9, doi: 10.1145/3055624.3075941.

[46] C. Kiourt, A. Koutsoudis, and G. Pavlidis, "DynaMus: A fully dynamic 3D virtual museum framework," *J. Cult. Herit.*, vol. 22, pp. 984–991, Nov. 2016, doi: 10.1016/j.culher.2016.06.007.

[47] N. Vidakis, A. Barianos, A. Trampas, S. Papadakis, M. Kalogiannakis, and K. Vassilakis, "Generating Education in-Game Data: The Case of an Ancient Theatre Serious Game," in *Proceedings of the 11th International Conference on Computer Supported Education*, 2019, pp. 36–43, doi: 10.5220/0007810800360043.

[48] N. Vidakis, A. K. Barianos, A. M. Trampas, S. Papadakis, M. Kalogiannakis, and K. Vassilakis, "in-Game Raw Data Collection and Visualization in the Context of the 'ThimelEdu' Educational Game," 2020, pp. 629–646.

[49] S. Papadakis, A. Trampas, A. Barianos, M. Kalogiannakis, and N. Vidakis, “Evaluating the Learning Process: The ‘ThimelEdu’ Educational Game Case Study,” in *Proceedings of the 12th International Conference on Computer Supported Education*, 2020, pp. 290–298, doi: 10.5220/0009379902900298.

[50] N. Vidakis, E. Syntychakis, K. Kalafatis, E. Christinaki, and G. Triantafyllidis, “Ludic Educational Game Creation Tool: Teaching Schoolers Road Safety,” in *Universal Access in Human-Computer Interaction. Access to Learning, Health and WellBeing*, Springer I., M. Antona and C. Stephanidis, Eds. 2015, pp. 565–576.

[51] U. Technologies, “Unity 3D Real-Time Development Platform.” https://unity.com/ (accessed May 16, 2020).

[52] J. K. Haas, “A History of the Unity Game Engine,” Worcester, 2014. [Online]. Available: https://core.ac.uk/download/pdf/212986458.pdf.

[53] “itch.io.” https://itch.io/ (accessed May 24, 2021).

[54] Valve, “Steam Store.” https://store.steampowered.com/ (accessed May 24, 2021).

[55] M. Toftedahl and H. Engström, “A Taxonomy of Game Engines and the Tools that Drive the Industry,” 2019.

[56] “NodeJS.” https://nodejs.org/ (accessed Apr. 13, 2021).

[57] Google, “V8 Engine.” .

[58] M. Satheesh, B. J. Dmellon, and J. Kron, *Web Development with MongoDB and NodeJS - Second Edition: Build an interactive and full-featured web application from scratch using Node.js and MongoDB*, Second. Birminghan: Packt Publishing Ltd, 2015.

[59] “ReactJS.” https://reactjs.org/ (accessed Jul. 15, 2021).

[60] A. Paudyal, “Developing Video Chat Application with ReactJs And WebRTC,” no. April, 2021.

[61] MongoDB, “MongoDB.” https://www.mongodb.com/ (accessed Apr. 13, 2021).

[62] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah, “A comparative study: MongoDB vs. MySQL,” in *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, Jun. 2015, pp. 1–6, doi: 10.1109/EMES.2015.7158433.

[63] Unity3D, “Unity Scripting Reference.” https://docs.unity3d.com/ScriptReference/index.html (accessed May 18, 2021).

[64] “Iolaos.” https://iolaos.nile.hmu.gr/ (accessed May 18, 2021).

[65] G. Bierman, C. Russo, G. Mainland, E. Meijer, and M. Torgersen, “Pause ’n’ Play: Formalizing Asynchronous C $^\\sharp$,” 2012, pp. 233–257.

[66] R. Reis, “TriLib 2.” https://ricardoreis.net/trilib-2/.

[67] NILE LAB, “NILE Lab.” https://nile.hmu.gr/en/home/ (accessed Sep. 01, 2021).

Appendix A - JSON Format Guidelines

General Format

For a game to be customizable, a JSON must be provided along with the game that will conform to the following general format. All JSON files applied must include a “fields” array where all fields needed for the form will be included. Settings is the second field, which a child named “levels enables”. This is a boolean that allows you to indicate that the game has different levels.

```
{
        "fields": [],
        "settings": {
                "levelsEnabled": false
        }
}
```

Available Fields

The following fields can be used to generate the form

String

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “string”
Title	String	Yes	The title of the field that will be presented to the user
Default	String	No	The default value of the field

```
{
        "name": "country",
        "type": "string",
        "title": "Country",
        "default": "Greece"
}
```

Number

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “number”
Title	String	Yes	The title of the field that will be presented to the user
Default	number	No	The default value of the field

```
{
        "name": "numSteps",
        "type": "number",
        "title": "Number of steps",
        "default": 10
}
```

Color

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “number”
Title	String	Yes	The title of the field that will be presented to the user
Default	String	No	The default value of the field

```
{
        "name": "hex",
        "type": "color",
        "title": "Color",
        "default": "#ff0000"
}
```

Enum

Presents the user with a dropdown menu, from which he/she can select from the predefined values.

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “enum”
Title	String	Yes	The title of the field that will be presented to the user
Default	Array<Item>	No	The default value of the field

The Item used in the Array has the following structure

Name	Type	Required	Description
title	String	Yes	The title of the field that will be presented to the user
value	String	Yes	The value the field will assume if the user makes this selection.

```
{
	"name": "car",
	"type": "enum",
	"title": "Car",
	"items": [
		{
			"title": "First car",
			"value": "car-1"
		},
		{
			"title": "Second car",
			"value": "car-2"
		},
		{
			"title": "Third car",
			"value": "car-3"
		}
	]
}
```

Choice

Allows the user to select a value from a predefined list. Will show checkboxes when multiple selections are possible and radio buttons when the selection is exclusive.

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “choice”
Title	String	Yes	The title of the field that will be presented to the user
multiple	boolean	Yes	Defines if selection is exclusive or multiple
Options	Array<Options>	Yes	The possible selections

The options have the following structure

Name	Type	Required	Description
Title	String	Yes	The title of the field that will be presented to the user
value	String	YEs	The value of the field
Selected	Boolean	No	Indicates whether this option is selected by the user

```
{
        "name": "foods",
        "type": "choice",
        "title": "Favorite foods",
        "options": [
                {
                        "title": "First food",
                        "value": "food-1",
                        "selected": true
                },
                {
                        "title": "Second food",
                        "value": "food-2",
                        "selected": false
                },
                {
                        "title": "Third food",
                        "value": "food-3",
                        "selected": true
                }
        ],
        "multiple": true
}
```

Image, Asset

Allows the user to upload images to our server.

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “image”
Title	String	Yes	The title of the field that will be presented to the user
multiple	boolean	Yes	Defines if multiple images can be uploaded
addItemPrompt	String	No	A prompt to be shown to users.
helpMessage	HelpMessage	No	A message that will be displayed to users, informing them about the file to be uploaded

```
{
        "name": "images",
        "type": "image",
        "title": "Images",
        "multiple": true,
        "addItemPrompt": "Add Image"
}
```

Object

Allows combinations of other available fields, creating more complex objects.

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “image”
Title	String	Yes	The title of the field that will be presented to the user
properties	Array<Field>	Yes	Array of fields included in the object

```
{
        "name": "color",
        "type": "object",
        "properties": [
                {
                        "name": "hex",
                        "type": "color",
                        "title": "HEX",
                        "default": "#0000ff"
                },
                {
                        "name": "colorName",
                        "type": "string",
                        "title": "Color Name",
                        "default": "Blue"
                }
        ]
}
```

Array

Allows multiple fields of the same type to be bundled

Name	Type	Required	Description
Name	String	Yes	The name of the field
Type	String	Yes	The type for the fields, should have the value “image”
Title	String	Yes	The title of the field that will be presented to the user
items	Field	Yes	One of the available field types
additemPrompt	String	No	A prompt to be shown to users.
Default	Array	No	An array of default values, based on array type

```
{
	"name": "names",
	"type": "array",
	"title": "Names",
	"items": {
		"type": "string",
		"title": "Name"
	},
	"default": ["John", "Jane"],
	"addItemPrompt": "Add Name"
}
```

Appendix B - JSON file for ThimelEdu

In order to utilize ThimelEdu from the front-end website and employ all functionality within the game, we had to create the JSON File from witch the dynamic form for content creation would be generated. This is the JSON created:

```
{
    "fields": [
        {
            "name": "texts",
            "type": "array",
            "title": "Texts",
            "items": {
                "name": "infoText",
                "type": "object",
                "properties": [
                    {
                        "name": "interestPoint",
                        "type": "enum",
                        "title": "Interest Point",
                        "items": [
                            {
                                "title": "Thimeli",
                                "value": "1"
                            },
                            {
                                "title": "Ekkyklima",
                                "value": "2"
                            },
                            {
                                "title": "Mechani",
                                "value": "3"
                            },
```

```
{
        "title": "Periaktos",
        "value": "4"
},
{
        "title": "Vrontio",
        "value": "5"
},
{
        "title": "Architecture",
        "value": "8"
},
{
        "title": "Skene",
        "value": "9"
},
{
        "title": "Proskenio",
        "value": "10"
},
{
        "title": "Logio",
        "value": "13"
},
{
        "title": "Orchestra",
        "value": "14"
},
{
        "title": "Evripos",
        "value": "15"
},
```

```
                {
                    "title": "Kilon",
                    "value": "17"
                },
                {
                    "title": "Diazoma",
                    "value": "18"
                },
                {
                    "title": "Klimakes",
                    "value": "19"
                },
                {
                    "title": "Kerkides",
                    "value": "20"
                },
                {
                    "title": "Edolia",
                    "value": "21"
                }
            ]
        },
        {
            "name": "infoTTitle",
            "type": "string",
            "title": "Informational Title",
            "default": "Θυμέλη"
        },
        {
            "name": "infoTText",
            "type": "string",
            "title": "Informational Text",
```

```
                "default": "An informative piece of information"
        },
        {
                "name": "infoTSchType",
                "type": "enum",
                "title": "School Type",
                "items": [
                        {
                                "title": "Mainstream School",
                                "value": "1"
                        },
                        {
                                "title": "Special School",
                                "value": "2"
                        }
                ]
        },
        {
                "name": "infoTLang",
                "type": "enum",
                "title": "Language",
                "items": [
                        {
                                "title": "English",
                                "value": "1"
                        },
                        {
                                "title": "Greek",
                                "value": "2"
                        }
                ]
        }
```

```
                ]
            },
            "addItemPrompt": "Add Informative Text"
        },
        {
            "name": "pictures",
            "type": "array",
            "title": "Pictures",
            "items": {
                "name": "infoPic",
                "type": "object",
                "properties": [
                    {
                        "name": "interestPoint",
                        "type": "enum",
                        "title": "Interest Point",
                        "items": [
                            {
                                "title": "Thimeli",
                                "value": "1"
                            },
                            {
                                "title": "Ekkyklima",
                                "value": "2"
                            },
                            {
                                "title": "Mechani",
                                "value": "3"
                            },
                            {
                                "title": "Periaktos",
                                "value": "4"
```

```
},
{
    "title": "Vrontio",
    "value": "5"
},
{
    "title": "Architecture",
    "value": "8"
},
{
    "title": "Skene",
    "value": "9"
},
{
    "title": "Proskenio",
    "value": "10"
},
{
    "title": "Logio",
    "value": "13"
},
{
    "title": "Orchestra",
    "value": "14"
},
{
    "title": "Evripos",
    "value": "15"
},
{
    "title": "Kilon",
    "value": "17"
```

```
                        },
                        {
                                "title": "Diazoma",
                                "value": "18"
                        },
                        {
                                "title": "Klimakes",
                                "value": "19"
                        },
                        {
                                "title": "Kerkides",
                                "value": "20"
                        },
                        {
                                "title": "Edolia",
                                "value": "21"
                        }
                ]
        },
        {
                "name": "infoPTitle",
                "type": "string",
                "title": "Informational Title",
                "default": "Thimele"
        },
        {
                "name": "infoPicture",
                "type": "image",
                "title": "Image",
                "multiple": false,
                "addItemPrompt": "Add Image"
        },
```

```
                {
                    "name": "infoTSchType",
                    "type": "enum",
                    "title": "School Type",
                    "items": [
                        {
                            "title": "Mainstream School",
                            "value": "1"
                        },
                        {
                            "title": "Special School",
                            "value": "2"
                        }
                    ]
                },
                {
                    "name": "infoTLang",
                    "type": "enum",
                    "title": "Language",
                    "items": [
                        {
                            "title": "English",
                            "value": "1"
                        },
                        {
                            "title": "Greek",
                            "value": "2"
                        }
                    ]
                }
            ]
        },
```

```
                "addItemPrompt": "Add Informative Picture"
        },
        {
                "name": "3D Objects",
                "type": "array",
                "title": "3D Objects",
                "items": {
                        "name": "interestObject",
                        "type": "object",
                        "properties": [
                                {
                                        "name": "interestPoint",
                                        "type": "enum",
                                        "title": "Interest Point",
                                        "items": [
                                                {
                                                        "title": "Thimeli",
                                                        "value": "1"
                                                },
                                                {
                                                        "title": "Ekkyklima",
                                                        "value": "2"
                                                },
                                                {
                                                        "title": "Mechani",
                                                        "value": "3"
                                                },
                                                {
                                                        "title": "Periaktos",
                                                        "value": "4"
                                                },
                                                {
```

```
            "title": "Vrontio",
            "value": "5"
        },
        {
            "title": "Architecture",
            "value": "8"
        },
        {
            "title": "Skene",
            "value": "9"
        },
        {
            "title": "Proskenio",
            "value": "10"
        },
        {
            "title": "Logio",
            "value": "13"
        },
        {
            "title": "Orchestra",
            "value": "14"
        },
        {
            "title": "Evripos",
            "value": "15"
        },
        {
            "title": "Kilon",
            "value": "17"
        },
        {
```

```
                                                "title": "Diazoma",
                                                "value": "18"
                                            },
                                            {
                                                "title": "Klimakes",
                                                "value": "19"
                                            },
                                            {
                                                "title": "Kerkides",
                                                "value": "20"
                                            },
                                            {
                                                "title": "Edolia",
                                                "value": "21"
                                            }
                                        ]
                                    },
                                    {
                                        "name": "interestObject",
                                        "type": "asset",
                                        "title": "Asset",
                                        "multiple": false,
                                        "addItemPrompt": "Add 3D Object"
                                    }
                                ]
                            },
                            "addItemPrompt": "Add 3D Object of Interest"
                        }
                    ],
                    "settings": {
                        "levelsEnabled": false
                    }
```

```
}
```