jamk

LMS integration SDK

Eetu Ihalainen

Bachelor’s thesis
November 2022
Bachelor's Degree Programme in Business Information Technology

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Eetu Ihalainen

LMS integration SDK

Jyväskylä: JAMK University of Applied Sciences, November 2022, 34 pages

Business, Degree Programme in Business Information Technology, Bachelor's Thesis.

Permission for open access publication: Yes

Language of publication: English

Abstract

Commissioner, Zaibatsu Interactive Oy, developed a learning simulation for Learning Management System platform. It was their first project of a sort, and it was done through trial and error.

Purpose of the action research was to create software development kit that allows developers to create efficiently simulations which can be integrated to the Learning Management System platforms.

As a result SCROM integration SDK was developed. The developed SDK enables developers to create simulations to the Learning Management System platforms more efficiently than before. Improvements to the building process were not as impactful as hoped but the start of a new project were noticeably more efficient than before.

Keywords/tags (subjects)

Unity 3D, LMS, SDK, SCORM, game development

Miscellaneous (Confidential information)

No confidential information was handled or gathered during the research

Contents

Figures

List of abbreviations

LMS	Learning Management System
SCORM	Shareable Content Object Reference Model
SDK	Software Development Kit
API	Application Programming Interface

1 Introduction

In early 2022 during my internship at Zaibatsu Interactive Oy, I got assigned to do a learning simulation for learning management system platform. Zaibatsu Interactive Oy and I had no previous experience doing learning simulations for learning management system platforms. Zaibatsu Interactive Oy pointed out some tools to help with the development process. Thus, the development process of the simulation was done in quick iterations and through trial and error. The simulation was done in Unity game engine as it was the main tool for making games at Zaibatsu Interactive Oy and likewise it was the game engine that I had the most experience in.

After the project I got assigned to do research on how to improve the development process of producing learning simulations for learning management system platform if similar projects were ever encountered in the future. Zaibatsu Interactive Oy is the commissioner of this research.

Zaibatsu Interactive Oy is game and software company founded in 2014 in Central Finland. Currently Zaibatsu has over 20 employees as well as a few interns. At Zaibatsu teams work on both game and software projects, either together with a small team or independently collaborating with client's team. On the software side, the main focus is on front-end development, while on the game side the focus is on free-to-play mobile games.

Zaibatsu Interactive Oy determined that producing a software development kit which contains everything necessary to create the learning simulations would be the approach for this research. The purpose of this research is to make the development process of the learning simulations for learning management systems as easy and effective as possible. The research questions were given by the commissioner after the earlier mentioned project was finished.

2 Research methods and research questions

The departure of the research is that commissioner, Zaibatsu Interactive Oy, has already produced a learning simulation to a learning management system platform. During the development of the first project some shortcomings were noticed. Purpose of the research is to help the development process of similar future projects.

2.1 Definition of objects

The research's goal is to make future development easier and faster with the intention of prototyping and producing a first playable demo as efficiently as possible. This research will give an answer to the research questions where the plan is to utilize already known tools and code as much as possible and customize those to be optimized specifically for WebGL and learning management system platforms. All these improvements will be placed in the SDK that will be used in the future development process.

2.2 Research questions

The research problem is how the development process can be more efficient. Thus, these following research questions were set by the commissioner.

1. What is always necessary for the SDK when targeting LMS platform?
2. How fast can a demo be produced?
3. How can the building process be speeded up?

2.3 Research method

The goal of the research is to develop and improve existing methods and produce a completely new SDK that is more efficient. Thus, the selected research method is action research. According to Rautio (2007), action research is self-directed methodology that aims to improve existing activity. The results should be compared to the targets of the research. The action research is not only development process but continuous learning process which consist of repetitive cycles, see figure 1 for the cycles. The cycles consist of action, evaluation, reflection and creating a model and planning the changes.

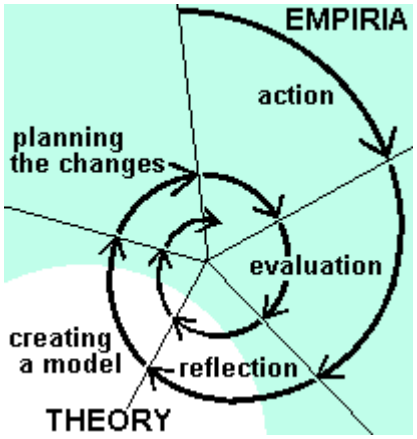

Figure 1 Repetitive cycles of action research (Rautio, 2007)

2.4 Information retrieval and source material

Manuals of different software are rapidly changing, and it is easier to publish changes and documentation to the internet rather than publish a book of it every time the documentation changes. For example, the user manual of the Unity game engine application programming interface (API) is on their own website and there is no published book of the manual. Thus, information in this research is mostly searched from the internet.

Keywords used to search for information were SDK, SCORM, LMS, Unity, Version Control, VCS.

3 Knowledge base

This chapter reviews the most important concepts and theoretical points of the thesis that are needed to create the software development kit. The chapter commences with the SDK then moving to concepts such as Unity, on which the SDK is built. Continuing to Learning Management System, which is the target platform and concluding on SCORM, which is the communication tool between the LMS platform and Unity WebGL.

3.1 Software development kit

Software development kit or SDK is a collection of software, tools, libraries, and programs that help developers with their development process. Typically, it is used to make programs for a targeted platform. For example, Android SDK Toolkit is made for developers that want to build Android mobile applications (CleverTap, 2020).

As earlier mentioned, the SDK is a collection of tools and software it is naturally intended to help with the development process. It is important that the SDK is as easy to use as possible and documented properly. Preferably the SDK should be working as soon as the SDK has been downloaded and installed (CleverTap, 2020).

3.2 Unity game engine

Unity is a real-time 3D development platform developed by Unity Technologies. Unity's main selling point is its ability to create video games and it is one of the leading mobile game development platforms, but it also has other use cases such as construction planning and simulation where the real-time rendering can be useful (Unity Technologies, 2021a; Unity Technologies, 2022a). While Unity is a 3D development platform it also has tools for 2D software development and many 2D games have been released with it (Unity Technologies, 2022b).

Unity has a large library of assets called Asset Store where tools for 3D or 2D development can be bought or sold. Asset Store contains a considerable amount of 3D models, visual effects, sound effects and Add-ons. Unity Assets Store also includes plenty of free assets (Unity Technologies, 2022c).

3.2.1 Unity Editor and interface

Unity Editor is the main application where developers create 2D or 3D games and applications (Unity Technologies, 2022d). Unity Editor has powerful graphical interface which helps with the

development process. It has many functionalities and is easily customizable (Unity Technologies, 2022e), see figure 2 for the Unity Editor interface.

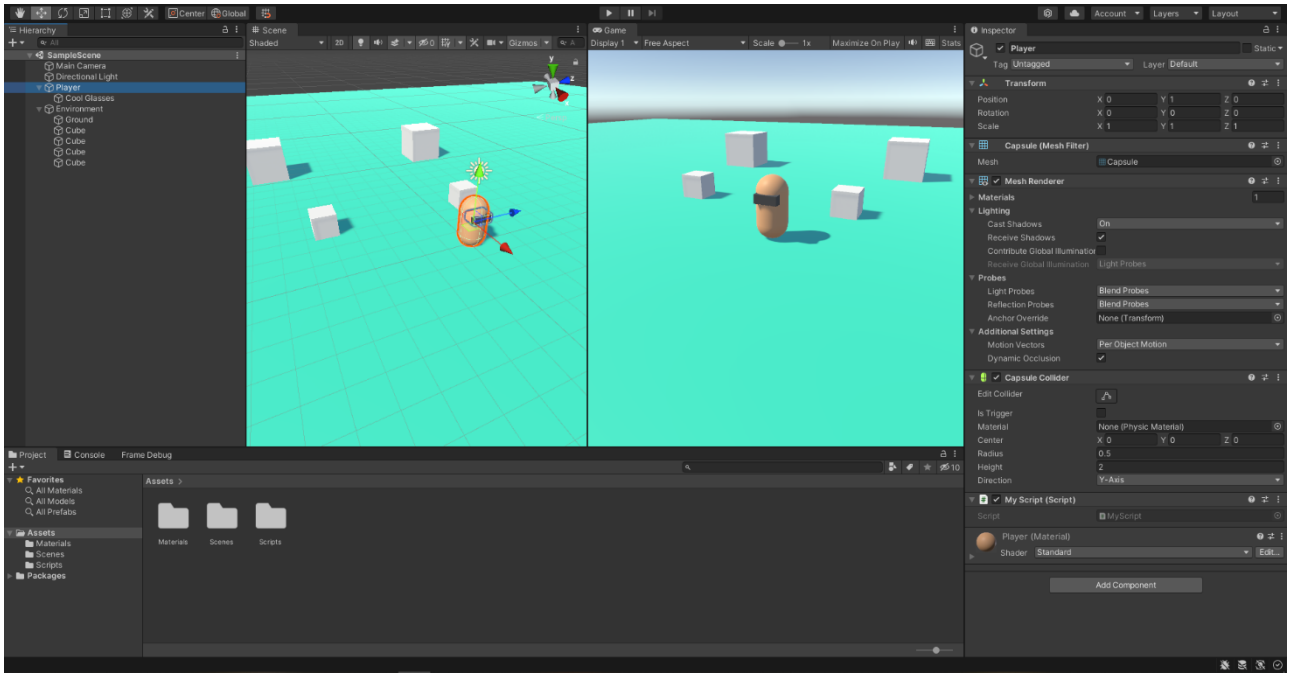

Figure 2 Unity Editor interface

In Unity, a scene is where most of the work is done. The scene holds most of the content in the application. Usually, the application is sliced into multiple scenes. Every scene could have its own rules or complexity. Developers can add functionalities to the scenes with game objects and components (Unity Technologies, 2022i). Game object is a fundamental piece or object that can represent anything. Game objects themselves do not much but they provide name, size, rotation, and scale of the object. Game objects hold components, and one game object can hold multiple components. Components add behaviors to the game objects and there are many ready components offered by Unity. It is possible to add custom components to the game objects with scripting (Unity Technologies, 2022f).

3.2.2 Scripting

Scripting is one of the essential concepts in development with Unity. Although Unity is mostly written in C++ language, scripting in Unity happens in C# programming language, which is natively supported. Other .NET programming languages can be used with a compatible dynamic link library.

Although Unity offers a lot of different functionalities, scripting can be used to create custom behavior and functionality, which are usually needed to create a modern application. Scripting can combine behaviors of existing Unity components, create whole new features, or even combine custom components (Unity Technologies, 2022f). See figure 3 for a simple script that prints *Hello World!* to the console. Occasionally, scripts that inherit MonoBehaviour are called as components.

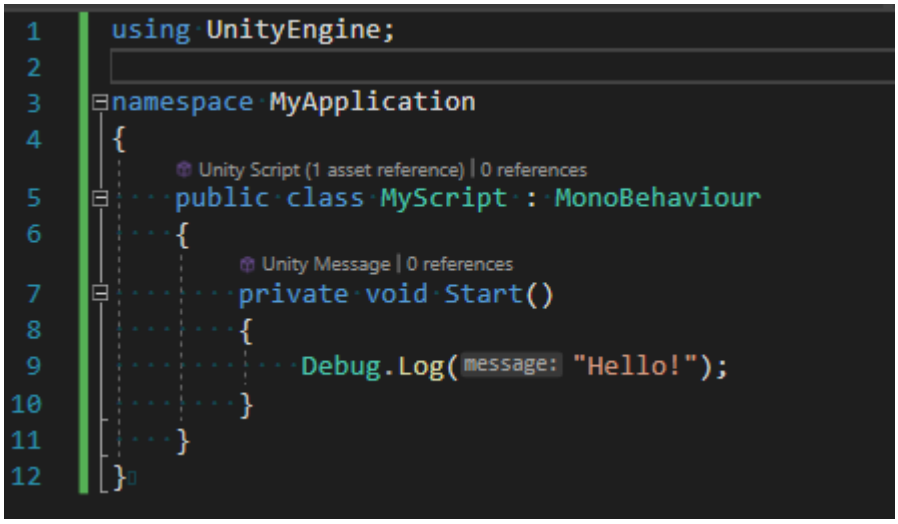

Figure 3 Script named *MyScript* which prints *Hello!* to the Unity Editors console.

3.2.3 Unity WebGL and publishing a build

Unity supports WebGL. WebGL is a JavaScript API (application programming interface) that allows 2D and 3D graphics to be rendered into a web browser (Unity Technologies, 2021b).

A build is the final product that will be used by the end user. Unity can publish a build for a variety of platforms, so it is important to check the current targeted platform. Targeted build platform can be chosen in build settings. There is also an option to choose what scenes should be included in the build (Unity Technologies, 2019a).

Unity has WebGL Templates that embeds the application in an HTML page so it can be played and shown in the browser. Custom templates can be added to create additional functionalities when the application is shown in the browser. Creating custom WebGL templates requires the developer to have prop knowledge of JavaScript programming language, concepts, and terminology (Unity Technologies, 2022j).

In Unity WebGL, there are limitations that must be considered, such as threading, memory management and audio. As Unity Technologies (2021b) states:

> *Threads are not supported due to the lack of threading supporting in JavaScript. This applies to both Unity's internal use of threads to speed up performance, and to the use of threads in script code and managed dlls. Essentially, anything in the System.Threading namespace is not supported (Platform support section)*

The lack of multithreading leads to audio problems and limitations. Normally Unity uses FMOD to manage audio, but it depends on multithreading. Thus, Unity WebGL has a custom backend for audio that lets the web browser manage playback and mixing (Unity Technologies, 2022g). Memory management should be considered when targeting WebGL platform since Unity heap tries to allocate contiguous blocks of memory. If browser fails to allocate that memory block, the application will crash. It is important to keep the heap as small as possible. Garbage collection works differently too. In WebGL garbage collector will only run when the stack is empty, which happens once a frame (Unity Technologies, 2022h).

3.3 Learning Management System

Learning management system or LMS is a software that manages educational content and provides a better learning experience for students and other learners (Valamis, 2019). Usually, LMS is a web-based platform where content providers such as schools can upload and modify courses and lessons. Learners and students can attend the lessons and the courses. Schools can monitor their progress and give grades depending on their performance. LMS is not just for schools, for example, it could be used to train new employees.

According to Valamis (2019), different LMS platforms have different features, but typically they offer the same core features. Commonly, the core features are user management, content management, score and performance management and ability to offer live and online seminar environments. Key benefits of LMS are easy access and ease of modifying the learning contents.

3.4 SCORM

SCORM stands for Sharable Content Object Reference Model developed by Advanced Distributed Learning, and it was first released in January 2000. It is a set of industry-standards that helps educational content and LMS platform to communicate with each other at run times. SCORM can track time, completion, pass/fail and overall score. SCORM is usually delivered to the LMS as a ZIP file. SCORM can be divided to smaller parts called SCO that can be imagined as chapter or modules. (Rustici Software, n.d.).

SCORM talks between the interface or activities and the LMS platform. For example, Activities or the interface informs SCORM to add points, pass or fail a course. Then SCORM communicates the information to the LMS platform at run time, meaning that it happens when the application is running.

There are multiple versions of SCORM and choosing the right version is important. Rustici Software (n.d.) points out that widely used versions such as SCORM 1.2 and SCORM 2004 are usually supported by the LMS platforms. The 2004 version has four editions, of which the fourth one is the most used. There are more modern versions available such as The Experience API, also known as xAPI, and it solves a lot of problems that the older versions have. One of the problems with older versions of SCORMs is that they can only be used in web browsers, but xAPI solves it. Although it solves problems, it is not as widely supported as the older versions.

3.5 Version Control

Version Control is a software tool that helps developers to manage changes to the source code or any files on a project. Version Control Systems enables tracing of every modification to the source code, this allows jumping back and forth to previous changes. Going back to the old changes helps when mistakes are made. (Atlassian, n.d.) It is considered as good practice to save changes, also know as pushing, as frequently as possible but doing so within the good taste. This is done because it is easier to revert the pushed mistakes.

In addition, version control enables developers to share code when version control platforms are used. Version control platform also helps save the project and making backups of the project. This

reduces the risk of losing the project when hardware failures happen. There are multiple version control platforms such as GitHub, GitLab and Bitbucket. There are plenty of version control systems available such as Git, Mercurial, CVS also known as Concurrent Versions Systems and Apache Subversion also known as SVN. According to Stack Overflow's annual developer survey (2022), over 93% of respondents used Git as their version control system.

Version control systems have immense value in production because they help developers to avoid conflicts during development. Version control system gives the ability to work freely and make changes in a good, organized way. (Atlassian, n.d.)

3.6 Relevance of knowledge base and concepts

These concepts come together while doing the implementation part of the research, where the development process of the SDK takes place. In the SDK, Unity acts as the foundation of the entire software development kit and other tools are added on top of it. Bitbucket's version control platform will be the location where the SDK will be saved.

4 Implementation

This chapter tells how to build the SDK down to every detail and setting. The chapter also talks about Zaibatsu SDK that is included in the SDK. This thesis does not go into the details of how to build the Zaibatsu SDK and what it specifically does but it gives a small overview of it. This chapter does not tell when to push changes to the version control since it should be as frequently as possible. There are some tools that are not available for free, but the chapter tells alternative free options available in the moment of writing.

4.1 Setting up the repository for the SDK

The project was started by cloning the Zaibatsu SDK repository**.** By calling `$git clone *URL of the project*` in the terminal it cloned the project by using HTTPS to the current directory. After cloning the Zaibatsu SDK, ties to the original repository had to be removed by deleting the git directory from the Zaibatsu SDK project's local directory. After ties to the original repository were

removed, a new repository had to be made, so that the new SCROM-integration-SDK has a base and version control.

Creating a new repository can be done in multiple ways and to several services. Good and free alternatives services are GitHub and Bitbucket. In this thesis Bitbucket was used because it is the service that Zaibatsu uses the most. By clicking *Create repository* from Bitbucket's frontpage it opens creating configurations for the new repository, see figure 4 for the configuration view. There is no need for git ignore file to be added during the creation of the repository. Git ignore file tells git what files to exclude from the source control. The reason to not include the git ignore file during the creation of the repository is because Zaibatsu SDK included this specific file that is configurated specifically for Unity. Otherwise, it is always a good idea to add git ignore file during the creation of the repository. This repository is private and no one else than Zaibatsu has access to it. By clicking *Create repository* creates the new repository with chosen configuration.

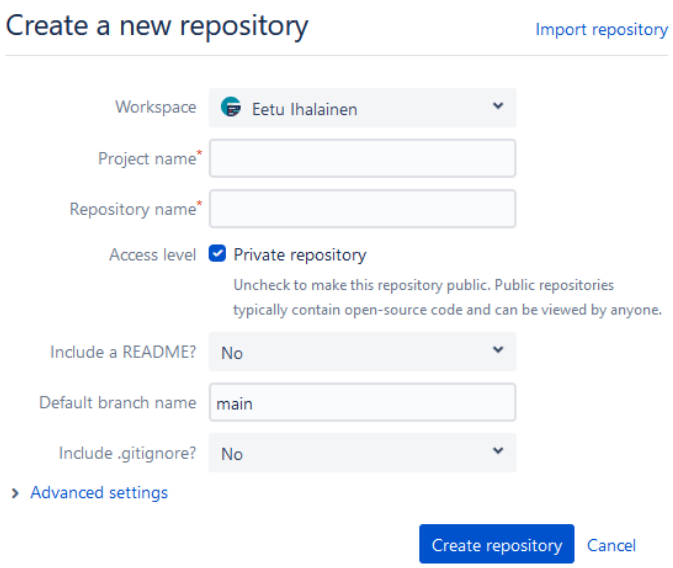

Figure 4 Bitbucket repository configuration view

Initializing local repository was done by switching to local directory of the project in the terminal as such `$cd C:/UnityProjects/SCORM-intergration-SDK`. Before connecting the local repository to the existing Bitbucket repository the current content of the project in the working tree had to be prepared for staging for the first commit. This was done by calling `$git add`. Furthermore, calling `$git add .` stages all the files expect the file types that are excluded by the git ignore file.

After staging, the content on the working tree was committed. Committing the staged files records the changes to the local repository. Committing was done calling `$git commit –m "message here"`. The -m option added a message to the commit which usually describes the contents of the commit. Connecting existing local repository to the Bitbucket was done by calling `$git remote add origin *URL of the repository here*`, this line tracks the Bitbucket repository or also known as remote with the given URL of the repository. Finally, the local working tree was pushed to Bitbucket repository by calling `$git push –u origin main`.

4.2 Setting up SCORM integration Kit

SCORM integration Kit allows Unity WebGL project to be integrated in LMS by SCORM (Stals, 2015). This specific integration tool was chosen because it was free of cost, and it completely fit with the needs of this software development kits.

4.2.1 Installing the SCORM integration Kit

SCORM integration Kit's package was downloaded from the SCORM integration Kit's website (https://unity3d.stals.com.au/scorm-integration-kit/) and then it was imported to the Unity project. Importing happened by clicking *Assets*, *Import package*, *Custom package* and then selecting the downloaded SCORM integration Kit -package and finally clicking *Import*.

4.2.2 Project set up

Several project settings had to be changed to get the WebGL and SCORM integration Kit to work. From build settings the platform had to be changed to WebGL. This was achieved by choosing the WebGL from the platforms list and clicking *Switch Platform*. See figure 5 for build settings. Note that *Switch Platform* only shows if other than current platform is selected.

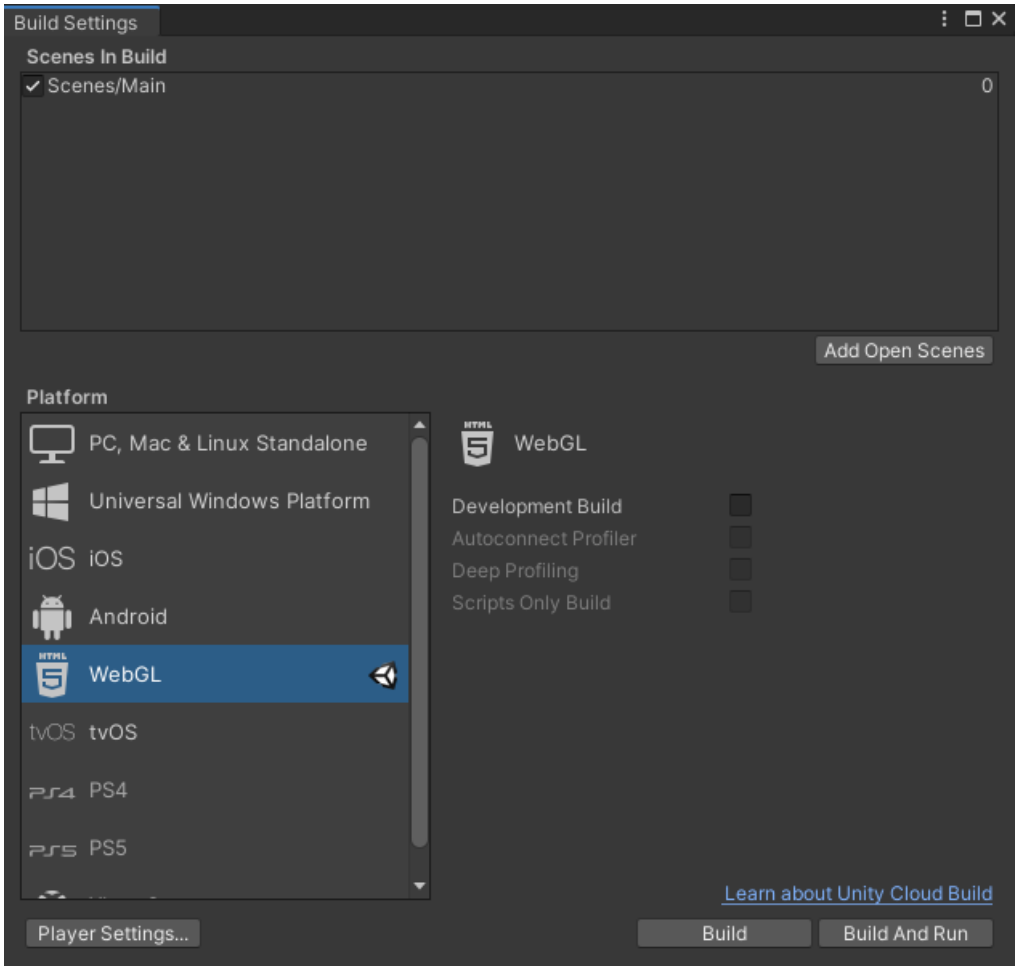

Figure 5 Build Settings

As earlier mentioned in the Unity WebGL and publishing a build section that to add extra functionalities to the WebGL application the WebGL templates are used. SCORM integration Kit offers custom Unity WebGL Template called SCORM. SCORM template contains scorm.js file that allows the communication to a SCORM API wrapper which acts as a messenger between Unity and the JavaScript code. See figure 6 for complete architectural design of the SCORM integration Kit. SCORM template was chosen from the player setting which can be accessed from the build settings.

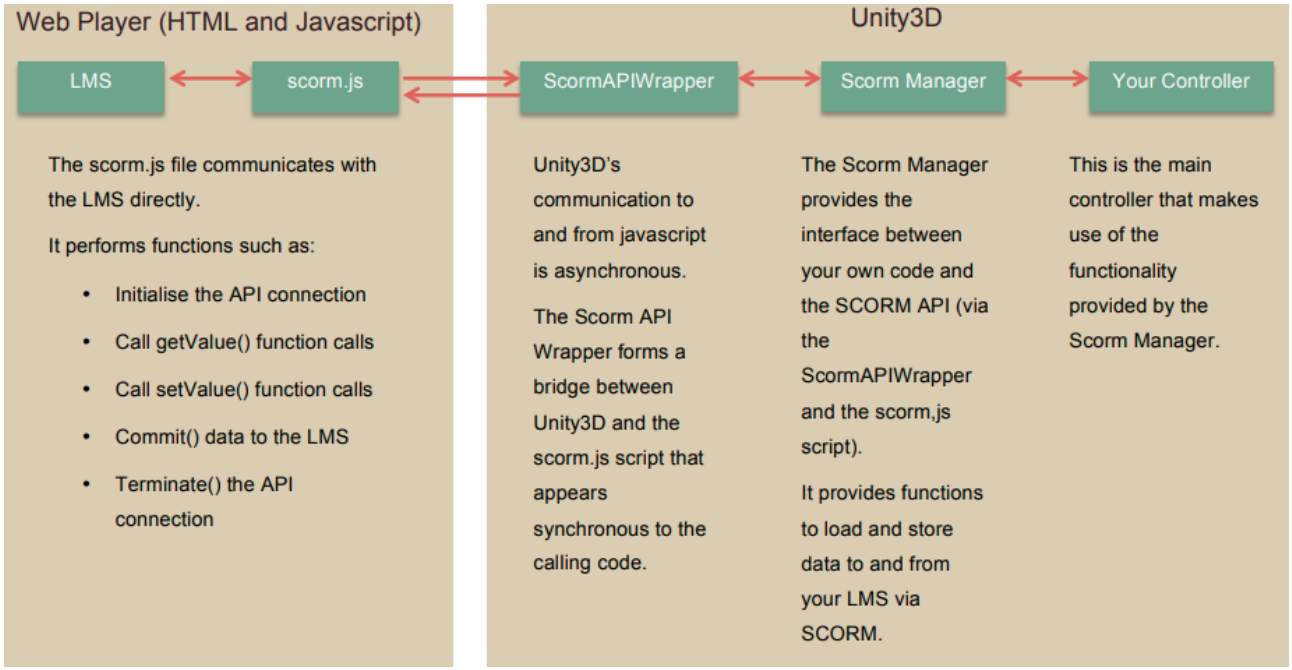

Figure 6 Architecture of SCORM integration Kit (Stals, 2015)

According to the Stals (2015) SCORM integration kit's manuals developers must create SCORM manager from the SCORM menu found in the toolbar or create a new game object which has script called *ScormManager.cs* attached to it. Furthermore, SCORM manager game object must have a child game object that has custom script which communicates with the SCORM manager. See figure 7 for the hierarchy of the scene.

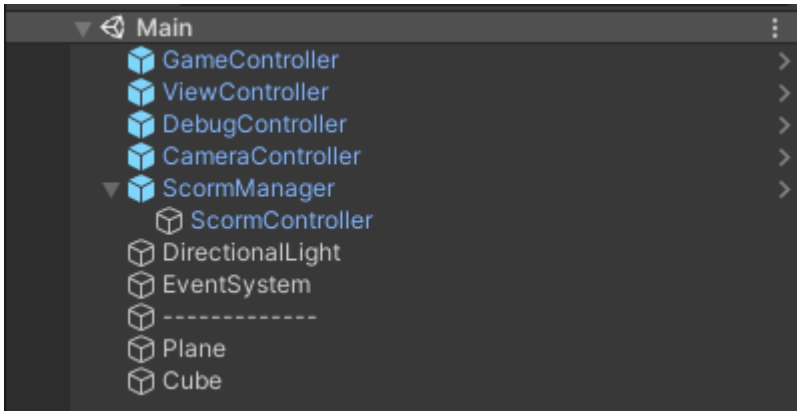

Figure 7 Demonstration project's hierarchy.

Custom SCORM Controller must include two important functions that initializes and terminates the SCORM. These functions are `Scorm_Initialize_Complete()` and `Scorm_Commit_Complete()`. These functions should be written explicitly as such because they are being called by the SCORM manager with Unity BroadcastMessage function which takes function name as an argument. Custom SCORM controller gives the ability to grade the SCO. See appendix 1 for the custom SCORM controller script.

4.3 Overview of the Zaibatsu SDK

Zaibatsu SDK is a software development kit for Unity game engine. It has been designed to enforce good coding conventions and to enable high code reusability between multiple Unity projects. Its core features are debugging tools, object pooling, state machine that can be used to visualize game flow and manage UI. Zaibatsu SDK allows developers to prototype and develop different types of games and simulations in a quick manner.

4.4 Demonstration project

During the development process of the SDK a simple demonstration project was developed, see figure 8. The purpose of this demonstration project was to see how the new SCORM-Integration-SDK would work in practice and to see that it works in LMS as well.

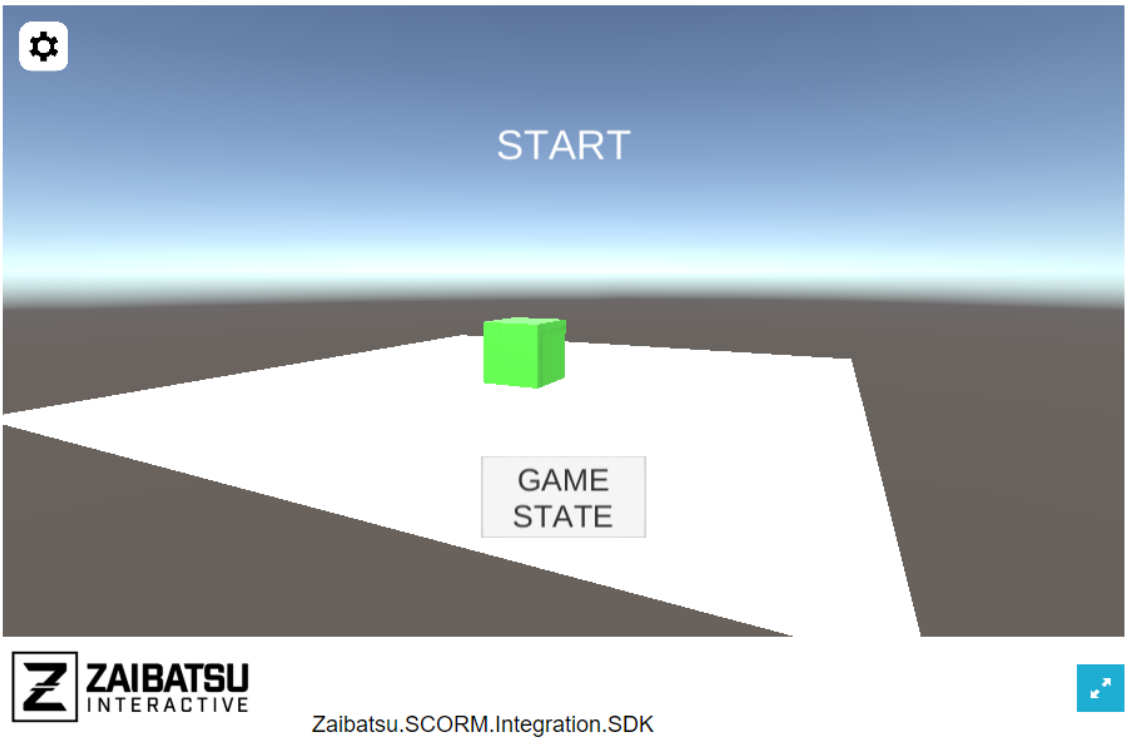

Figure 8 Demonstration project in Moodle LMS

The demonstration project has “pass or fail” grading with unlimited attempts and only the highest graded attempt will be saved to the LMS. The project has four simple states, start state, game state, lose state and score state. Game starts with the start state which has a button that leads to the game state. Respectively game state has two buttons where lose button leads to the lose state and score button leads to the score state. Entering the lose state, the student fails the course with a score of 0. As earlier stated, the game has unlimited tries, and the student can try the test course again. If a student enters the score state, the student gets a score of 100, which is the maximum

score, and the course is passed.

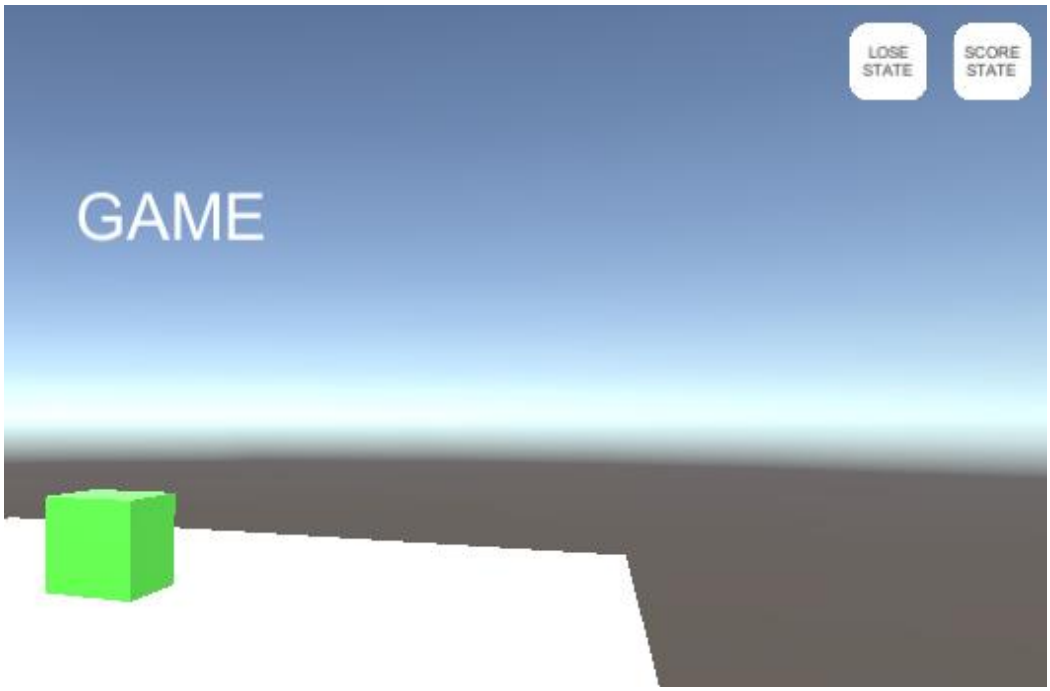

Figure 9 Lose and Score state button seen on the top right corner.

4.5 Building and importing SCORM package to LMS

Learning Management System platform used to test this demonstration project's SCORM package was Moodle Sandbox (https://sandbox.moodledemo.net). Generally, the LMS platforms are nearly identical about importing the SCORM packages follows the same steps in every LMS platform.

First, the application had to be built from the build settings, see figure 5. for the build settings. Before building the application scenes in the build was selected from the *Scene In Build* list. After pressing the *Build* and selecting the destination directory the build process started. A few minutes later the build process had finished, and the build process of the application was ready.

Next step was to publish the built application as SCORM package. SCORM Export window was opened from the toolbar under *SCORM* and *Export SCORM package*, see figure 10 for the SCORM exporting window. SCORM Export window has multiple settings to configurate. Few important settings are folder location, time limit, and completed by measure. Folder location is the directory where the application was built. This had to be done manually either searching through the directories or by writing the exact directory path. Completed by measure should be enabled if progress

of the SCO is the only graded measure to pass the SCO. Time limit tells how much time the learner has to complete the SCO. In this case it set to unlimited.

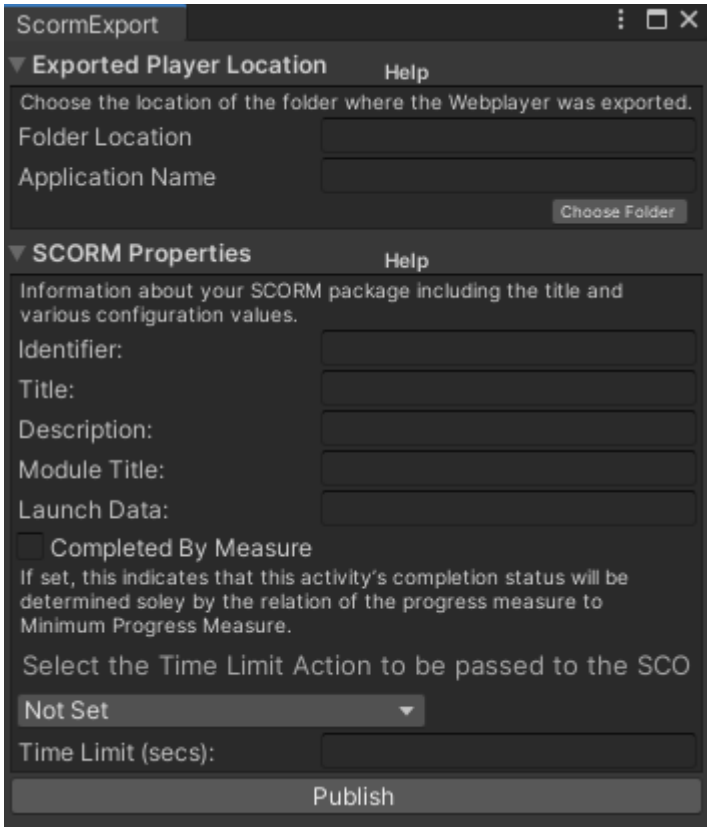

Figure 10 SCORM Export window

After filling the SCORM export windows configurations, publish button on the bottom was pressed to publish the SCORM package as ZIP file to the selected directory.

A new course to the LMS platform was needed for the SCORM package as the Moodle Sandbox did not have any. This was achieved by clicking *Add a new course* from my courses section. Adding the new course to the LMS might vary depending on the platform. SCORM package was added to one of the course topics as an activity. This happened by clicking the desired topic and editing it. Activities generally want a file, and, in this case, it was the SCORM package, see figure 11 and figure 12 for the package insertion to the LMS platform.

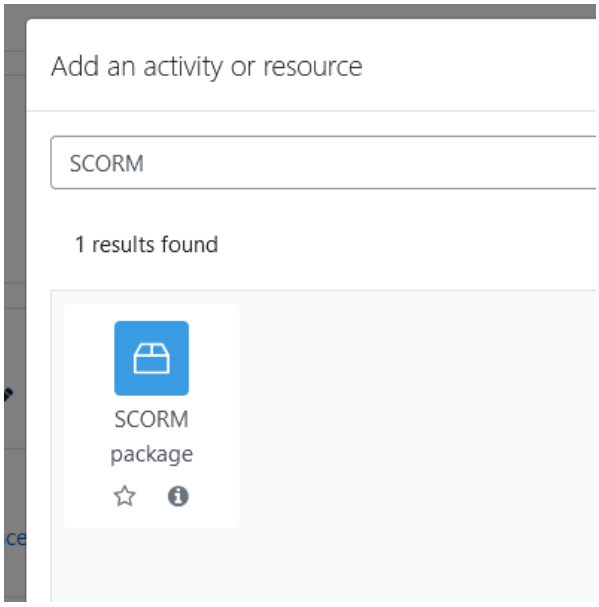

Figure 11 Package being added to the LMS

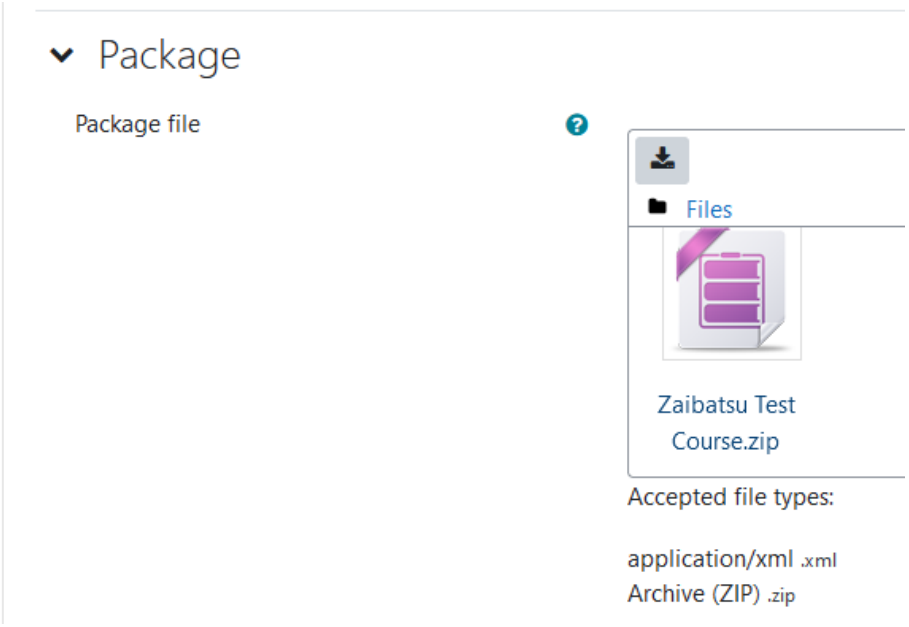

Figure 12 SCORM package seen in the topic settings

LMS platform may possibly get the grading setting from the SCORM package, but it is a good idea to ensure that it is set as preferred. After SCORM package was set to the LMS and topic settings

were saved the demonstration project was ready to be tested in the LMS platform. See figure 8 for the demonstration project in the LMS and figure 13 for the graded attempt of the project.

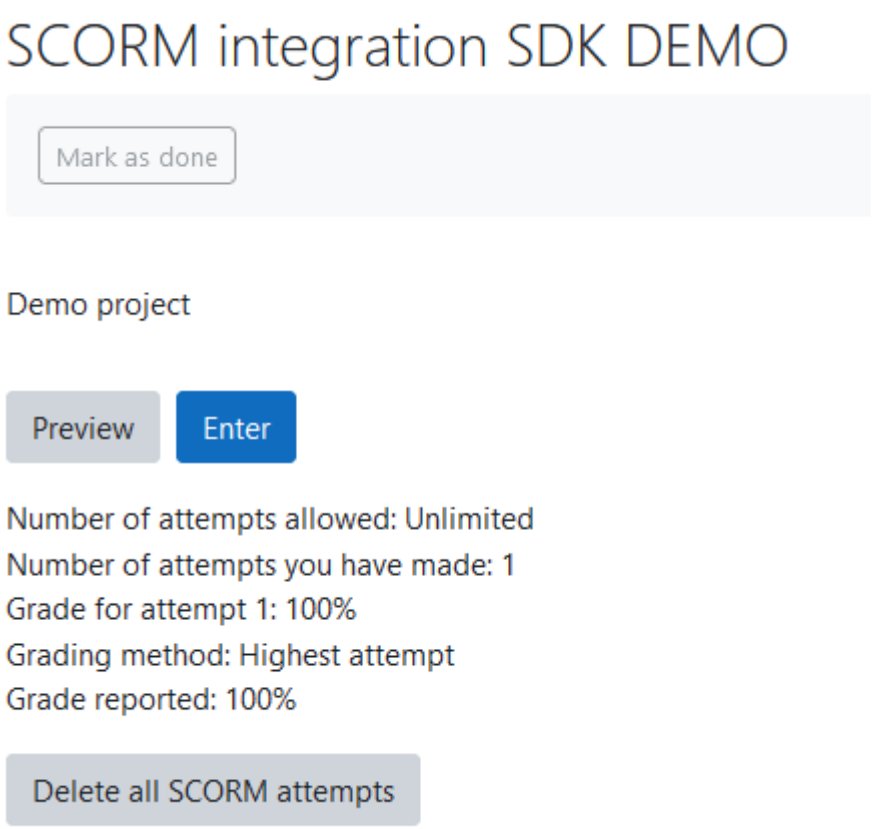

Figure 13 Demonstration project after successful attempt

4.6 Improving the workflow

During the earlier development shortcomings were noticed when building the application to be ready for the SCORM publishing. Main problem with it was that developers had to search for the files manually after building the application. Searching the files manually can be found cumbersome and tedious. To address this the SCORM publishing was modified to be done automatically after the build was done.

The variables of the SCORM export windows were saved with PlayerPrefs so they were simple to modify from the outside of the SCORM export window. A new way to manipulate was introduced since the folder selection was not needed anymore. ScriptableObject based approach was taken.

According to Unity (2022k) ScriptableObject is a script that acts as data container that can be saved as an asset. Therefore, the new export settings can be easily accesses from the project assets, see figure 14 for the new ScriptableObject SCORM export settings.

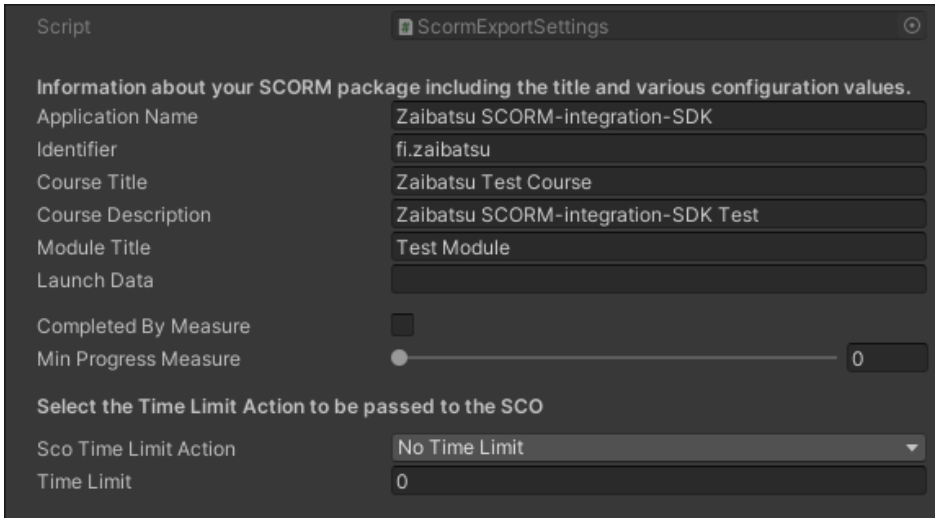

Figure 14 ScribtableObject SCORM export settings

The automatic publishing of the SCORM package after build process was achieved by creating a so-called post processing script that gets notified after the build process has been done. The post processing script takes the build path of the application and saves the build path to correct PlayerPref called *Course_Export*. After saving the build path the `ScormExport.Publish()` function is called which takes the settings and exports the SCORM package as ZIP file. See the appendix 2, for the implementation of the BuildPostProcess.cs.

5 Results and conclusions

This chapter goes into answering the research questions, results, conclusions, and what future development could hold.

5.1 Answering research questions

5.1.1 What is always necessary on the SDK when targeting LMS platform?

Earlier mentioned in the 3.1 SDK section that SDK is a collection of tools and programs. Therefore, without the tools and programs the SDK wouldn't be useful, and the core tool of this SDK is the SCORM integration Kit. Without the kit this SDK wouldn't be working a solution. The Zaibatsu SDK added to the newly developed SDK is not necessary but it allows faster development for that reason the SDK is a good addition. More tools could be added to the SDK, but the size of the project should be kept as minimal as possible since the target is Unity WebGL and it has memory limitations as earlier mentioned in the section 3.2.4. For that reason, no tools should be added prematurely to the SDK and should only be added when project in hand required the tools to be added.

5.1.2 How fast can a demo be produced?

With the Zaibatsu SDK a demo can be produced within hours or minutes depending on the size and complexity of the simulation. The demonstration project was developed after the SDK was done and the development time for it was under an hour. The demonstration project is incredibly simple hence the development time was under an hour.

5.1.3 How can the building process be speeded up?

At the start the building process was manual and it had to be done in two separate steps. The building process was found tedious due the searching of the file twice manually. During the development the building process was slightly improved as the two step process got stripped to just one step.

At the start of the development the first step was to build the application and then second step was to export the built application as SCORM package. During the configuration of the SCORM package the applications build directory had to be searched manually.

After the development the first and only step was to build the project and then the it would ask the name of the SCORM package ZIP file. The build process is still manual and could take tens of

minutes depending on the size of the built application. This could be improved with future development of DevOps pipeline, but it was out of the scope of this research.

5.2 Results

The research problem was how the development process could be more efficient. As a result, working SCORM integration SDK was developed which makes the development process more efficient. The SDK helps from the start of the project to the moment of publishing the project as SCORM package, see figure 15 for the steps how the development process goes with the SDK. Since the action based research method required that the results should be compared against the target of the research, it shows that the SCORM integration SDK works as the target of the research was the development process should more efficient than before.

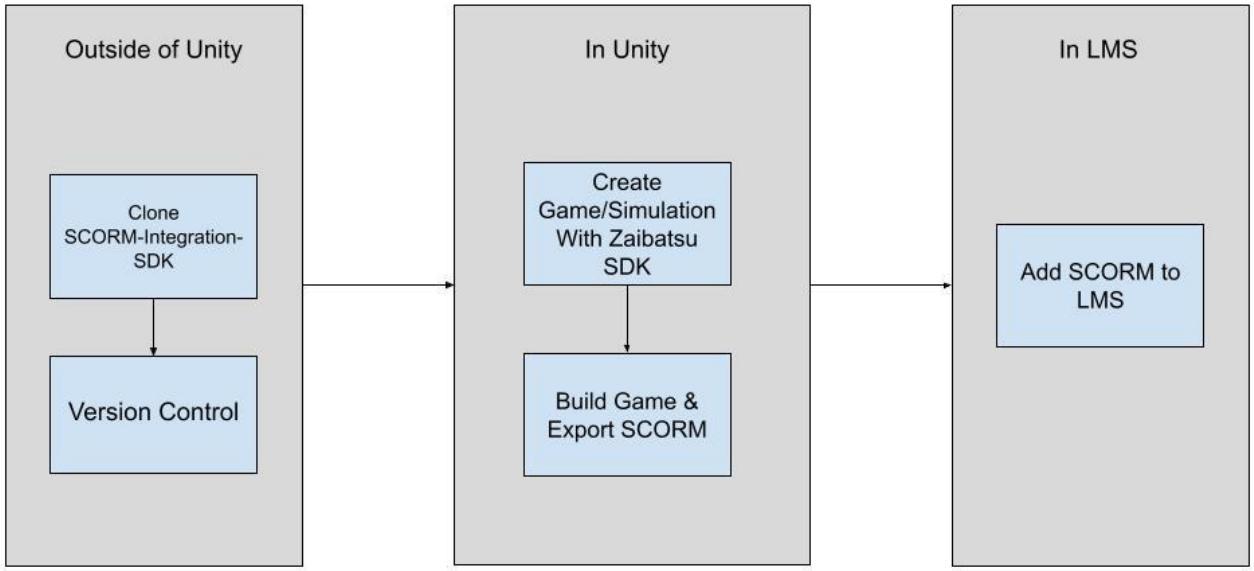

Figure 15 Steps to create a simulation to a LMS platform with the developed SDK

5.3 Conclusions

Setting up the SCORM integration Kit, new Unity project for WebGL and source control was the most time-consuming part of the development. Then SCORM integration Kit's custom controller was the second most time-consuming part.

It is possible that the chosen SCORM integration Kit was not the most ideal solution, as other alternative solutions were not explored in depth. For example, Unity Asset Store has alternative SCORM integration solution that might be worthwhile to investigate in future projects. This is one thing that could have been done differently. The SCORM integration Kit did not follow the known industry best practices on the Unity C# scripting side. The naming conventions of the kit and uses of BroadcastMessage functions made the kit confusing to use. Also the SCORM integration Kit comes with major deficit since it does not work on newer releases of the Unity game engine. Current highest supported Unity engine release is the newest 2019.4 long term support release.

The improvements of the building process were not as impactful as hoped. Since the building process takes place on the developer local computer. The building process blocks the developer to continue the development process during building. One option for this could be DevOps pipeline but it out of the scope of this research.

As a conclusion the developed SCORM integration SDK improves the early stages of the development in greatly manner but later on the development the building process might still be time-consuming depending on the size of the project. By comparing the figures 15 and 16 the improvements can be seen visually.

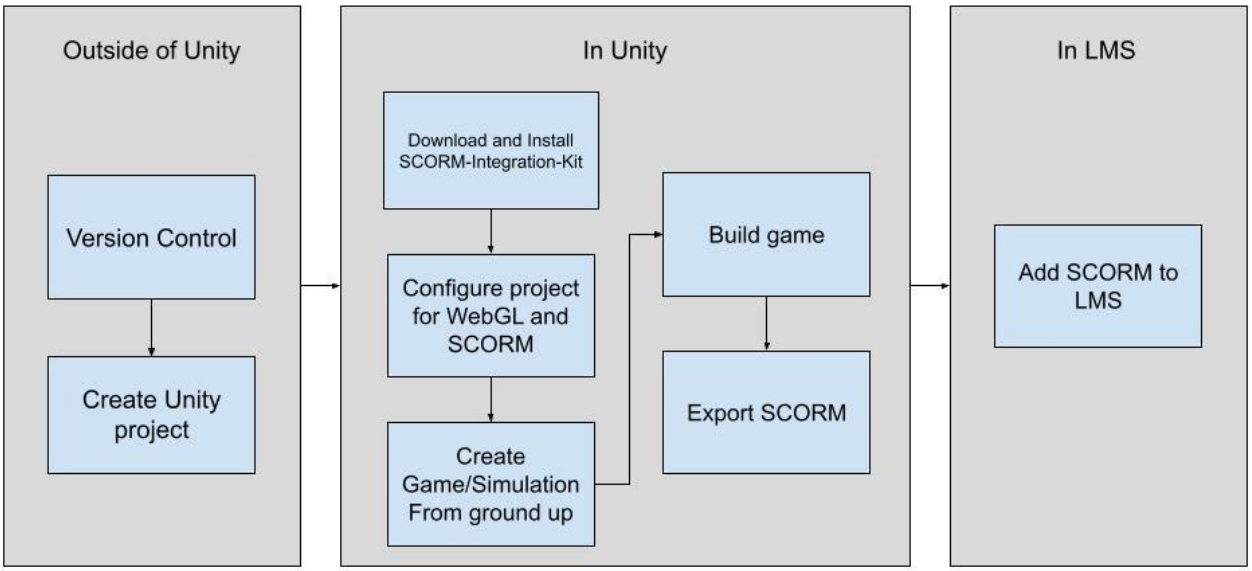

Figure 16 Steps to create a simulation to a LMS platform without the developed SDK

5.3.1 Further development

The build process could benefit from future research because in the current implementation as the project grows so does the build process time. This prevents the developer to work while the building process in ongoing as the Unity Editor cannot be used at the same time. Future development could look into how to build a DevOps pipeline to deliver the SCORM packages when the changes are noticed in the version control.

Future development could also research how to take different platform such as mobile devices into account. The current implementation does not take mobile devices into account at all since the current WebGL builds do not support mobile devices.

6 Discussion

6.1 Reliability of the research and sources

As mentioned before in the information gathering section the information was mainly gathered from different internet sources. The sources for programs were original manuals of the programs so the gathered information would be accurate. Information that was not related to the software were gathered from recently published sources and authors backgrounds were checked to ensure that the authors would have experience in the field. All the steps can be reproduced and tested by others, excluding the usage of Zaibatsu SDK. These factors rise the reliability of the research.

On the other hand, some might question reliability of the research based on the fact that all the cycles of the action research were not mentioned explicitly. It could have been better if all the cycles would have mentioned individually. Due to lack of time the count of cycles was minimal.

6.2 Ethics review

No confidential information was handled or gathered during the research. Research permit was not needed during the research. Plagiarism is avoided by using references and sourced appropriately according to the reporting style.

References

CleverTap. (2020). What is an SDK? Everything you need to know. Retrieved August 6, 2022. https://clevertap.com/blog/what-is-an-sdk/

Rautio, P. (2007) Action Research. Retrieved November 19, 2022. http://www2.uiah.fi/projects/metodi/120.htm

Rustici Software. (n.d.). Evolution of SCORM. Retrieved August 5, 2022. https://scorm.com/scorm-explained/business-of-scorm/scorm-versions/

Stack Overflow. (2022) Developer Survey 2022. https://survey.stackoverflow.co/2022/#education-ed-level

Stals, R. (2015). SCORM Integration Kit. Retrieved November 2, 2022. https://unity3d.stals.com.au/scorm-integration-kit/

Unity Technologies. (2019a). Creating and Using Scripts. Retrieved August 5, 2022. https://docs.unity3d.com/Manual/BuildSettings.html

Unity Technologies. (2021a). Our Company. Retrieved August 5, 2022. https://unity.com/our-company

Unity Technologies. (2021b). Getting started with WebGL development. Retrieved August 5, 2022. https://docs.unity3d.com/2018.4/Documentation/Manual/webgl-gettingstarted.html

Unity Technologies. (2022a). Real-time 3D technology in AEC, explained. Retrieved August 5, 2022. https://unity.com/solutions/architecture-engineering-construction/rt3d-explained

Unity Technologies. (2022b). Real-time rendering in 3D and 2D. Retrieved August 5, 2022. https://unity.com/how-to/real-time-rendering-3d

Unity Technologies. (2022c). Quick guide to the Unity Asset Store. Retrieved August 5, 2022. https://unity3d.com/quick-guide-to-unity-asset-store

Unity Technologies. (2022d). Unity User Manual. Retrieved August 5, 2022. https://docs.unity3d.com/2021.3/Documentation/Manual/UnityManual.html

Unity Technologies. (2022e). Unity's interface. Retrieved August 5, 2022. https://docs.unity3d.com/2021.3/Documentation/Manual/UsingTheEditor.html

Unity Technologies. (2022f). Creating and Using Scripts. Retrieved August 5, 2022. https://docs.unity3d.com/2022.2/Documentation/Manual/CreatingAndUsingScripts.html

Unity Technologies. (2022g). Audio in WebGL. Retrieved August 5, 2022. https://docs.unity3d.com/Manual/webgl-audio.html

Unity Technologies. (2022h). Memory in WebGL. Retrieved August 6, 2022. https://docs.unity3d.com/2019.4/Documentation/Manual/webgl-memory.html

Unity Technologies. (2022i). Scenes. Retrieved August 6, 2022. https://docs.unity3d.com/2022.2/Documentation/Manual/CreatingScenes.html

Unity Technologies. (2022j). Using WebGL Templates. Retrieved November 19, 2022. https://docs.unity3d.com/2019.4/Documentation/Manual/webgl-templates.html

Unity Technologies. (2022k). ScriptableObject. Retrieved November 19, 2022. https://docs.unity3d.com/Manual/class-ScriptableObject.html

Valamis. (2019). What is an LMS? Retrieved August 5, 2022. https://www.valamis.com/hub/what-is-an-lms

Appendices

Appendix 1. ScormController.cs

```
using UnityEngine;

public class ScormController : SingletonBehaviour<ScormController>
{
    private const float MIN_SCORE = 0;
    private const float MAX_SCORE = 100f;

    private float stopTime => Time.time - startTime;
    private float startTime;

    /// <summary>
    ///  Scorm_Initialize_Complete gets called by ScormManager after scorm is
initialized.
    /// </summary>
    public void Scorm_Initialize_Complete()
    {
        InitializeScorm();
    }

    /// <summary>
    ///  Scorm_Commit_Complete gets called by ScormManager after scorm is com-
mit has been called
    /// </summary>
    public void Scorm_Commit_Complete()
    {
        ScormManager.Terminate();
    }

    /// <summary>
    /// Handles course passed with score. Commits score to the LMS and exits.
    /// </summary>
    public void HandleCoursePassed()
    {
        CoursePassed();
        ExitAndCommit();
    }

    /// <summary>
    /// Handles course failed with score. Commits score and exits
    /// </summary>
    public void HandleCourseFailed()
    {
        CourseFailed();
        ExitAndCommit();
    }


    /// <summary>
    /// Initializes the SCORM
    /// </summary>
    private void InitializeScorm()
    {
        startTime = Time.time;
        ScormManager.SetProgressMeasure( 0f );
```

```
        ScormManager.SetScoreMin( MIN_SCORE );
        ScormManager.SetScoreMax( MAX_SCORE );


        if (ScormManager.GetCompletionStatus() != StudentRecord.CompletionSta-
tusType.completed)
        {
            ScormManager.SetCompletionStatus( StudentRecord.CompletionSta-
tusType.incomplete );
        }

        if (ScormManager.GetSuccessStatus() != StudentRecord.SuccessSta-
tusType.passed)
        {
            ScormManager.SetSuccessStatus( StudentRecord.SuccessStatusType.un-
known );
        }
    }

    /// <summary>
    /// Passes the course with MAX_SCORE
    /// </summary>
    private void CoursePassed()
    {
        SetProgressWithScore( 1f, MAX_SCORE );
        ScormManager.SetCompletionStatus( StudentRecord.CompletionSta-
tusType.completed );
        ScormManager.SetSuccessStatus( StudentRecord.SuccessStatusType.passed
);
    }

    /// <summary>
    /// Fails the course with MIN_SCORE
    /// </summary>
    private void CourseFailed()
    {
        SetProgressWithScore( 1f, MIN_SCORE );
        ScormManager.SetCompletionStatus( StudentRecord.CompletionSta-
tusType.incomplete );
        ScormManager.SetSuccessStatus( StudentRecord.SuccessStatusType.failed
);
    }

    /// <summary>
    /// Sets the learners progress and score
    /// </summary>
    /// <param name="progress">Learners progress</param>
    /// <param name="score">Learners score</param>
    private void SetProgressWithScore(float progress, float score)
    {
        ScormManager.SetProgressMeasure( progress );
        ScormManager.SetScoreRaw( score );
        ScormManager.SetScoreScaled( score / MAX_SCORE );
    }

    /// <summary>
    /// Exits the SCORM and Commits
    /// </summary>
    private void ExitAndCommit()
    {
```

```
        ScormManager.SetSessionTime( stopTime );
        ScormManager.SetExit( StudentRecord.ExitType.normal );
        ScormManager.Commit();
    }
}
```

Appendix 2. BuildPostProcess.cs

```
#if UNITY_EDITOR
using UnityEditor;
using UnityEditor.Callbacks;
using UnityEngine;

public class BuildPostProcess
{
    [PostProcessBuild]
    public static void OnPostProcessBuild(BuildTarget buildTarget, string
buildPath)
    {
        PlayerPrefs.SetString("Course_Export", buildPath);
        ScormExport.Publish();
    }
}
#endif
```