Tampereen yliopisto

Topi Nieminen

UNITY GAME ENGINE IN VISUALIZATION, SIMULATION AND MODELLING

Faculty of Information Technology and Communication Sciences
Bachelor's thesis
July 2021

ABSTRACT

Topi Nieminen: Unity game engine in visualization, simulation and modelling
Bachelor’s thesis
Tampere University
Bachelor's Degree Programme in Computer Sciences
July 2021

Many software projects include different visualisation, simulation and modelling techniques. These can be used to represent data in human-comprehensible form, allow testing in simulations, and model complex or large objects in virtual environments to help design. Many academic research projects require these techniques as well. Game engines are software frameworks that implement these features to speed up development. This thesis investigates the usability of the Unity game engine for research projects. This is done via a systematic literature review on case studies that use Unity in their projects. The thesis found that Unity can be used in a wide variety of research fields and there were some common advantages and challenges of using Unity in the case studies. The most common advantages of Unity were its ease of use, licensing and scripting support, while challenges included the need of file format conversions, limited documentation for research projects and lacking simulation performance.

Keywords: Unity, Unity3D, game engine, visualization, simulation, modelling, virtual reality

TIIVISTELMÄ

Topi Nieminen: Unity pelimoottori visualisoinnissa, simuloinnissa ja mallintamisessa
Kandidaatintutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Heinäkuu 2021

Pelimoottorit ovat ohjelmistokehyksiä, jotka on rakennettu edesauttamaan videopelien kehittämistä. Pelimoottori yleisesti toteuttaa muun muassa tietokonegrafiikoiden piirtämisen näytölle, fysiikkasimulaation sekä kommunikaation laitteiston ja käyttöjärjestelmän kanssa. Pelimoottori täten helpottaa ja nopeuttaa videopelin kehittäjän työtä, kun tämä voi keskittyä enemmän itse pelin toteuttamiseen. Pelimoottorin tarjoamat hyödyt eivät kuitenkaan rajoitu yksistään videopelien kehitykseen, vaan niistä voi olla hyötyä muissakin sovelluskohteissa. Tämän työn tarkoituksena oli selvittää, miten Unity, suosittu pelimoottori, soveltuu visualisointiin, simulointiin ja mallinnukseen akateemisessa kontekstissa.

Työ toteutettiin systemaattisena kirjallisuuskatsauksena akateemisiin tapaustutkimuksiin. Avainsanoista muodostettu kysely lähetettiin kolmeen akateemiseen tietokantaan, joiden tuloksista saatiin 16 työn aiheeseen liittyvää tapaustutkimusta. Tapaustutkimuksia rajattiin vielä kolmella kriteerillä: tapaustutkimuksen tuli olla kirjoitettu englannin kielellä, tapaustutkimuksen piti kuvata visualisointi-, simulointi- tai mallinnusprojektia, ja tapaustutkimuksen projektin tuli perustua suurilta osin Unity-pelimoottorin käyttöön. Tämä pienensi kirjallisuuskatsauksen joukkoa lopulliseen 12 tapaustutkimukseen.

Työn tulokset osoittavat, että Unity on käyttökelpoinen usealla tieteenalalla: kirjallisuuskatsaukseen sisältyi tutkimuksia bioinformatiikan-, tekniikan-, informaatiotieteiden- sekä arkkitehtuurin ja ympäristösuunnittelun aloilta. Tapaustutkimuksista etsittiin ja analysoitiin kommentteja Unityn käyttökelpoisuudesta, jotka voitiin luokitella hyötyihin ja haasteisiin. Yleisimmät hyödyt liittyivät muun muassa Unityn helppokäyttöisyyteen, lisensointimalliin ja ohjelmointirajapintaan. Yleisimpiä haasteita olivat Unityn rajoittunut tiedostomuototuki, lähes yksinomaan videopelikehitykseen keskittyvä dokumentaatio ja kehittäjäyhteisö, sekä tiettyihin käyttökohteisiin riittämätön suorituskyky.

Vähiten haasteita Unityn käyttökelpoisuudessa oli mallinnukseen liittyvissä tapaustutkimuksissa, joihin sisältyi pääasiassa arkkitehtuurin ja ympäristösuunnittelun alojen tutkimukset. Nämä haasteet liittyivät pääasiassa tiedostomuototukeen ja mallinnustyökalujen koordinaatistoon. Visualisointiin liittyvissä tutkimuksissa oli jonkin verran samoja haasteita kuin mallinnuksessa, mutta niiden lisäksi osa projekteista kohtasi suorituskykyongelmia. Eniten haasteita Unityn suorituskyky toi projekteihin, jotka ajoivat simulaatioita. Kaiken kaikkiaan Unity oli käyttökelpoinen lähes kaikissa tapaustutkimuksissa, tai tarjosi ainakin huomionarvoisen, joskin rajoittuneen, vaihtoehdon joidenkin tieteenalojen tavanomaisille työkaluille.

.Avainsanat: Unity, Unity3D, pelimoottori, visualisointi, simulointi, mallintaminen

1 Introduction

Visualisation, simulation and modelling techniques are required in many software projects. Visualisation techniques allow representing the underlying software processes in human-comprehensible form, while simulations allow imitating the real world in different ways. Complex objects or large cities can be modelled in a virtual environment to facilitate better design and planning. Academic research takes use of these techniques as well: data must be visualized, experiments can be done in simulations that sometimes take place in a carefully modelled virtual environment. Many applications and software frameworks have been developed to implement these techniques. One such class of software are video game engines.

Video game engines are software frameworks that have been created to support video game development. They often implement software features required in video games, such as graphics rendering, physics simulation, and simple modelling tools usually in the form of a level editor. Having these features already implemented simplifies the development process allowing the developer to focus on features specific to the project at hand. Since game engine features are not required only in video games, they could be used for other projects, such as research.

This thesis aims, through the methods of a literature review, to investigate the advantages and challenges of using a video game engine in academic projects. The game engine chosen for this is Unity, which is a popular game engine developed by Unity Technologies. Section 2 introduces the research and data collection methods used, as well as an overview of the case studies reviewed in this thesis. Section 3 provides an overview of Unity as a development tool, its software ecosystem and licensing. Section 4 expands on Section 2 by categorizing the studies and providing small synopses for each case study. Section 5 categorises and discusses the different advantages and challenges of using Unity in research projects, as found in the case studies. Finally, Section 6 discusses the results and some of the limitations of this thesis.

2 Research method

The research method used in this thesis is systematic literature review. Systematic literature reviews are a type of literature review that collects secondary data from other studies using systematic and replicable methods [Armstrong 2011]. Systematic reviews formulate research questions to identify relevant studies. The initial research question for this thesis was of form ”Usage of the Unity game engine outside video game development”. The data was to be collected from both academic and non-academic sources. However, the initial search results indicated that Unity is widely used in academia, spe-

cifically in visualization, simulation, and modelling. This lead to the current form of the research question, ”Unity game engine in visualization, simulation, and modelling”.

2.1 Data collection

The material for this literature review was searched for and collected from three sources: ProQuest, ScienceDirect and Tampere University’s ANDOR search engine. The following search query was used for all the databases: *(”Unity” OR ”Unity3D” OR ”Unity game engine” OR ”game engine”) AND (”Visualization” OR ”Simulation”).* The searches were done on 23.9.2019.

The databases were set to sort according to relevancy, and material was collected from first-page results only to keep the scope of this thesis restricted. Three rules were followed when evaluating relevancy: (1) the material must be written in English, (2) the material must address the Unity game engine and (3) the material must describe a case study on visualization, modelling or simulation done using Unity. Initially 16 documents were found relevant but this was later narrowed down to 12 using the criteria mentioned earlier.

Table 1 lists all the case studies examined in this literature review. Included of each case study are the authors, year, title and name of the publication. The differences in publications show that usage of Unity was not limited to particular fields or type of research. The oldest study is from the year 2012 and the latest ones from 2019. Studies from 2017 to 2019 represent the majority. This could be explained either by the search query favouring newer studies, the date of the search, or that Unity has simply become more popular from 2017 onwards.

Author(s)	Year	Title	Publication
Ismail Buyuksalih, Serdar Bayburt, Gurcan Buyuksalih, A.P. Baskaraca, Hairi Karim, and Alias Rahman	2017	3D modelling and visualization based on the Unity game engine – advantages and challenges	ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences
David Fredrick	2014	Time.deltaTime: the vicissitudes of presence in visualizing Roman houses with game engine technology	AI & SOCIETY: Journal of Knowledge, Culture and Communications
Bernhard Klein	2016	Managing the scalability of visual exploration using game engines to analyse UHI Scenarios	Procedia Engineering
Dany Laksono and Trias Aditya	2019	Utilizing A Game Engine for Interactive 3D Topographic Data Visualization	International Journal of Geo-Information
C. Masato Nakano, Erick Moen, Hye Suk Byun, Heng Ma, Bradley Newman, Alexander McDowell, Tao Wei, and Mohamed Y. El-Naggar	2016	iBET: Immersive visualization of biological electron-transfer dynamics	Journal of Molecular Graphics and Modelling
Minh Nguyen, Mohammed Melaisi, Brent Cowan, Alvaro Joffre Uribe Quevedo, and Bill Kapralos	2017	Low-end haptic devices for knee bone drilling in a serious game	World Journal of Science, Technology and Sustainable Development
Jiri Polcar, Petr Horejsi, Pavel Kopecek, and Muhammad Latif	2017	Using Unity3D as an Elevator Simulation Tool	DAAAM International Symposium on Intelligent Manufacturing and Automation
Azarakhsh Rafiee, Pim Van der Male, Eduardo Dias, and Henk Scholten	2017	Developing a wind turbine planning platform: Integration of “sound propagation modeleGIS-game engine” triplet	Environment Modelling & Software
Erick Ratamero, Dom Bellini, Christopher Dowson, and Rudolf Römer	2018	Touching proteins with virtual bare hands	Journal of Computer-Aided Molecular Design
Jue Wang, Keith Bennett, and Edward Guinness	2012	Virtual Astronaut for Scientific Visualization – A Prototype for Santa Maria Crater on Mars	Future Internet
Michael Wiebrands, Chris Malajczuk, Andrew Woods, Andrew Rohl, and Ricardo Mancera	2018	Molecular Dynamics Visualization (MDV): Stereoscopic 3D Display of Biomolecular Structure and Interactions Using the Unity Game Engine	Journal of Integrative Bioinformatics
Jimmy Zhang, Alex Paciorkowski, Paul Craig, and Feng Cui	2019	BioVR: a platform for virtual reality assisted biological data integration and visualization	BMC Bioinformatics

Table 1: Studies included in the systematic literature review.

3 Unity

3.1 General overview

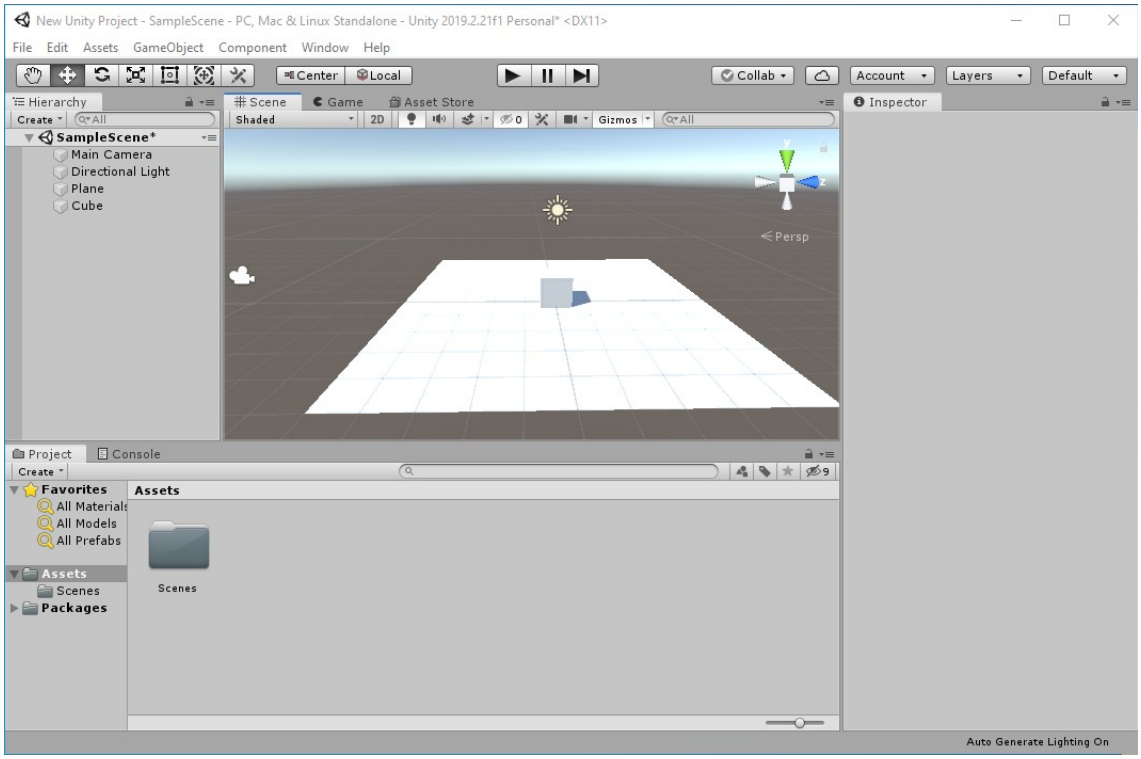

Figure 1: Unity editor user interface (version 2019.3).

Unity applications are created using the Unity editor shown in Figure 1. The Unity editor is available to the main desktop operating systems, Windows, MacOS and Linux (Ubuntu 16.04). Unity applications themselves can be deployed on desktop operating systems mentioned before, as well as Android- and iOS- mobile operating systems, PlayStation 4 and Xbox One consoles, and as a web application via WebGL. [Unity Technologies, 2021d] The cross-platform nature of Unity allows sharing much source code between the different platforms. This helps in saving costs when porting Unity applications to other platforms to increase customers or user base.

3.2 Workflow and ecosystem

Unity editor allows building applications using *scenes*. Scenes can include different 3D models, textures, and application logic via scripts. The base building blocks in a scene are called *game objects*. A game object can be thought of as a container that can hold the different 3D models and other assets. A game object can be modified by adding *components* to it. Components are most commonly scripts that alter the behaviour of the game object. Examples of components are, for example, a *RigidBody* component that simulates gravity on the game object, and the *Collider* component that lets the developer add collision detection to the game object. Multiple game objects and their com-

ponents can be combined into *prefabs*, which are reusable between different scenes, and can be instantiated into the scene from scripts. [Unity Technologies, 2021d]

The developers can create scripts themselves using the C# programming language. Developer-made scripts are the main way to add custom logic to the game or application being developed. Unity uses the open-source .NET platform with scripting back-ends to allow the usage of C# in Unity scripting [Unity Technologies, 2021d]. To enable the C# scripts to communicate with the Unity engine they must inherit Unity's built-in base class, *MonoBehaviour*. MonoBehaviour provides various functions and attributes necessary to interact with the engine.

Besides MonoBehaviour -derived scripts, developers can also include code created outside Unity via *plug-ins*. The plug-ins can be further divided to *Managed plug-ins* and *Native plug-ins*. Managed plug-ins are C# assemblies compiled outside Unity scripts. However, they are accessible to the standard .NET tools that Unity uses to compile scripts. Therefore their usage is similar to default Unity components. Native plug-ins are platform-specific code libraries that can access features such as operating system calls and third-party code that would otherwise be unavailable to Unity. Unity's tools have less access to these types of plug-ins, for example, when debugging code. [Unity Technologies, 2021d]

Noteworthy aspects of Unity are the *Unity Asset Store* [Unity Technologies, 2021j] and *Package Manager* [Unity Technologies. 2021d]. The asset store is a marketplace where third parties can sell resources for application development or add-ons to the Unity editor itself. Examples of development resources include assets such as 3D models, audio effects, scripts or templates, while editor add-ons add plug-ins and new tools to the editor interface. Unity's package manager can be used to download and install packages similar to the asset store, such as tools. By default the package manager downloads packages from a repository managed by Unity Technologies but community repositories can be accessed with it as well [Unity Technologies. 2021d].

3.3 Licensing

Unity applications are royalty free software, and Unity Technologies does not charge payments from software made using the editor [Unity Technologies, 2021h]. The creator of the software made with the Unity editor has all the intellectual, among other, rights to the software [Unity Technologies, 2021i]. However, earning money by selling the software made with the Unity editor has some conditions. Publishing software made with Unity requires a license from Unity Technologies [Unity Technologies, 2021f]. Unity Technologies offer four tiers of licenses to users that are listed in Table 2. Each license has a price per user, also known as a *seat*, and revenue requirements for the employing organization. The most notable license here is the Unity personal license, which

is free for individuals and small teams earning less than $100 000 in the preceding 12 months [Unity Technologies, 2021f].

Name	**Price per seat**	**Allowed revenue in last 12 months**
Personal	Free	Less than $100 000
Plus	$399 per year	Less than $200 000
Pro	$1800 per year	More than $200 000
Enterprise	$200 per month	More than $200 000

Table 2: Unity license pricing tiers. Data from Unity Technologies [2021f].

In addition to the licences offered to for-profit organizations, Unity Technologies offers a free Unity education license to not-for-profit academic institutions for uses such as education and research. This license can also be used by students of the academic institution. This license must be applied for and must meet the qualification requirements for the license program. These requirements differ by region. [Unity Technologies 2021g]

The Unity Education license appears the most suitable for research. The case studies discussed in this systematic literature review mostly reported having used the personal license, however. One explanation for this could be that the case studies were completed when the licensing terms were different.

4 Case study categorization

This section introduces the case studies included in this systematic literature review via providing outlines of the studies. The studies were grouped into three categories: Bioinformatics, Architecture and environmental modelling, and Engineering and information sciences. The studies were categorised by comparing their subject matter, fields of study and the academic publication they appeared in, such as a scientific journal or conference. Table 3 shows the result of the categorization.

Bioinformatics	Architecture and environmental modelling	Engineering and information sciences
Nakano and others [2016]	Buyuksalih and others [2017]	Klein [2016]
Ratamero and others [2018]	Fredrick [2013]	Nguyen and others [2017]
Wiebrands and others [2018]	Laksono and Aditya [2019]	Polcar and others [2017]
Zhang and others [2019]	Rafiee and others [2017]	
	Wang and others [2012]	

Table 3: Case studies categorized in three categories.

4.1 Bioinformatics

The Bioinformatics category contains studies related to the visualization, simulation and modelling of biological data, such as proteins and other complex molecules.

Nakano and others [2016] present iBET, an extension to VizBET, a computational framework to simulate and visualize electron-transfer dynamics in biological molecules. They argue that visualization in 2D on traditional computer monitors, that VizBET offers, severely limits user's ability to understand complex, multidimensional electron-transfer pathways. To provide a more accurate representation with enhanced depth perception, they created iBET that allows visualization using *virtual reality* (VR). Nakano and others [2016] describe procedures to export Visual Molecular Dynamics software models into Unity and render them in VR to be used with a *head-mounted display* (HMD). VR is a method of visualisation where graphics are rendered to a HMD to simulate the user being in a virtual environment.

Ratamero and others [2018] introduce a software pipeline that allows embedding protein structures into VR programs using a combination of widely available software, standard hardware and few custom-built scripts. The software pipeline uses Unity as the programming and execution environment. The researchers argue that three-dimensional models are more accessible and intuitive than 2D projections.

Wiebrands and others [2018] have developed a biomolecular visualization tool to be used with HIVE Cylinder, a 3 meters high 180 degree cylindrical 3D screen at Curtin University, Perth, Western Australia. Wiebrands and others explain that the HIVE Cylinder is very similar to a HMD but supports audiences up to 50 people. They argue that collaborative experiences with this system is easier than using multiple headsets.

Zhang and others [2019] present BioVR, a VR-assisted platform for visualisation and analysis of DNA, RNA, and protein sequences and structures. The paper describes the development of the project such as software design and user interface considerations, and creation and integration of different software components. The project aims to augment the field of bioinformatics in both analysis and visualization in virtual reality.

4.2 Architecture and environmental modelling

The studies belonging to this category describe projects that include, to an extent, elements from architecture and environmental modelling. These studies used different technologies to help virtualize real-life environments and objects. The virtualized environments were then brought to Unity for further use.

Buyuksalih and others [2017] describe advantages and challenges of using Unity in 3D modelling and visualization of two projects: the solar energy estimation system and underground utility mapping for Istanbul. The paper describes the development stages of the projects and discusses advantages and limitations of 3D modelling in Unity.

Fredrick [2013] describes the creation of a virtual model of a Pompeian house at the time of the eruption of Vesuvius. The house model was created using Unity by six undergraduate students. The students had access to plans, drawings, photographs and other documentation of the original house.

Laksono and Aditya [2019] explored the use of Unity for visualizing large-scale topographic data. The data were from mixed sources of terrestrial laser scanner models and topographic map data. The result was a multiplatform 3D visualization application that allowed its users to explore 3D environments in first person view or from a top-down view. The 3D visualization was build into WebGL and deployed to a web-server to be used from a web browser.

Rafiee and others [2017] described a conceptual design and implementation of a software meant to support wind turbine planning process. The software is an integration of three major components: Unity, geographic information system (GIS) and sound model. The software is aimed at the general public. Rafiee and others argue that the easy-to-use interface of the software allows contribution from users with differing technical backgrounds. They add that that interactivity provided by Unity has a potential to attract contribution from younger citizens as well.

Wang and others [2012] cover their approach and implementation of the Virtual Astronaut (VA) visualization environment built on Unity. Wang and others describe the purpose of the project was to support the scientific visualization of multiple-mission data from Mars. The VA allows the user to explore a virtual environment created with this data.

4.3 Engineering and information sciences

The three studies in the engineering and information sciences use Unity mainly for simulation of environment phenomena and mechanical devices, including the relevant user interactions.

Klein [2016] studied the feasibility of using visual pre-processing on heat emission data from cars using cellular automata for simulation and Unity for visualization. Klein describes the objective of the study to find out how domain knowledge can help reduce data and thus increase performance. Klein reported data reduction rate of 90% in model/file size without loss of visual quality using this approach.

Nguyen and others [2017] examine the use of consumer-level haptic devices as an affordable training solution for medical-based, surgical skills development. They develop a serious game in VR to simulate surgical bone drilling using Unity.

Polcar and others [2017] developed an elevator simulation model using Unity. Polcar and others then evaluated the model in terms of accuracy and suitability as a tool for creating an elevator simulation with a sequence dispatcher. They then compared the model to discrete event simulation (DES) tools that are usually used in industrial processes and discussed the advantages, disadvantages and limits of using such a model.

5 Advantages and challenges of Unity

This section introduces advantages and challenges of using Unity in academic projects collected from the case studies. Most studies did not explicitly mention advantages or challenges. Thus, data collection required interpretation. There were three ways the case studies addressed the advantages and challenges. Firstly, some studies clearly explained their experiences with Unity, usually as a list of advantages and challenges. These studies required very little interpretation and the information could be used as-is. Secondly, most studies explained the reasons why Unity was chosen for the case study. The reasons were interpreted as advantages of Unity. Thirdly, some studies had remarks towards some parts or features of Unity that could be interpreted as positive or negative feedback. These were interpreted as advantages and challenges, respectively.

5.1 Advantages

The case studies shared some common themes in their advantages. This allowed categorisation. Table 4 shows which advantage categories are applicable to each case study. The next sections will provide more details on the different categories.

	Accessibility and licensing	Built-in features	Cross-Platform deployment	External assets & plugins	File format support	Scripting	VR capabilities
Buyuksalih et al. [2017]	x	(x)	x		x	x	
Fredrick [2014]	x	x	x				
Klein [2016]	x	x				x	
Laksono & Aditya [2019]		x	x	x		x	
Nakano et al. [2016]	x	(x)		x	x	x	x
Nguyen et al. [2017]	x	x		x		x	
Polcar et al. [2017]	x	x				x	
Rafiee et al. [2017]	x	(x)	x			x	
Ratamero et al. [2018]	x	x		x	x	x	x
Wang et al. [2012]	x	x	x		x	x	
Wiebrands et al. [2018]		(x)		x		(x)	
Zhang et al. [2019]	x	(x)		x		(x)	x

Table 4: Advantage categories interpreted from case studies. ‘x’ means the case study found the advantage, ‘(x)’ means the advantage was not mentioned in-text but can be interpreted in other ways (e.g. context), and empty cell means the study did find advantages in that category.

5.1.1 Accessibility and licensing

This category combines attributes related to the accessibility, usability, cost and licensing of Unity. As seen in Table 4, at least one of these attributes was commented on in most sources. Fredrick [2014] states that Unity was chosen for the facts that Unity is free for academic use and because of accessibility of the game engine. Nakano and others [2016] reason that since Unity's personal license is free, it is an attractive option for scientists to incorporate into their research. Rafiee and others [2017] state that they use the personal license. Polcar and others [2017] describe Unity as a cheaper alternative compared to using traditional discrete-event simulation tools in elevator simulations. It can be assumed that Polcar and others refer to Unity's licensing as mentioned by Fredrick and Nakano and others when discussing the cost of Unity.

Several studies commended Unity's usability. Nguyen and others [2017] state that they were able to use Unity for rapid prototyping in their study. This is echoed by Ratamero and others [2018] who state that, as a game engine, Unity allows them to implement ideas quickly and easily. On the same note, Klein [2016] concludes that game engines provide powerful tools to easily construct urban scenes and provide first person exploration techniques for multidimensional data. Polcar and others [2017] reason that knowledge of C# and basic overview of Unity's scripting API were enough to create the model used in their study. Wang and others [2012] state that they found it easy and fast to implement functions with Unity.

Unity's learning resources were commented on by two studies. Buyuksalih and others [2017] state that Unity has numerous video tutorials and examples that help developers get familiar with the Unity development process, while Zhang and others [2019] state Unity's ample online resources as one of the reasons for choosing Unity for their project.

5.1.2 Built-in features

This category focuses on the built-in features of Unity. These include the general game engine components such as graphics, audio, scene-construction, as well as Unity's editor features. It can be safely assumed that all studies in this literature review used them to some extent, even if not directly mentioned. For example, Klein [2016] states that Unity engine was chosen for visualization because of the several built-in functions that support urban heat island exploration and multifaceted scene construction. Ratamero and others [2018] found that Unity's built-in features allow people without programming experience to develop their ideas with minimal training.

The UI systems that are included with Unity were mentioned in two studies. Laksono and Aditya [2019] built menus with Unity UI components, and added scripts to control their interactions. Wiebrands and others [2018], using standard Unity UI components with mouse pointer input, built a UI designed to work on both flat screen monitors and in stereo on large screen cylindrical displays.

The graphical features of Unity received some praise. Buyuksalih and others [2017] praised the engine's visualization and lighting of 3D objects arguing it as an advantage in modelling 3D spatial geometry. Polcar and others [2017] noted that simulations made with Unity can be made to look photorealistic. They also argued that Unity can reach much more realistic simulation than abstract DES tools. Laksono and Aditya [2019] also see game engines, including Unity, as an effective way to provide simulated environments.

Ratamero and others [2018] noted that Unity allowed them to attach colliders and rigid-body physics to protein objects. This allowed them to simulate virtual mass and thus deepen the physical reality simulation to the user.

The appreciation for other features of Unity were little more scattered. Zhang and others [2019], and Nakano and others [2016] mentioned Unity's virtual reality capabilities as a major advantage.

5.1.3 Cross-platform deployment

Five studies reported having built their software on more than one platform. Buyuksalih and others [2017] state that the ability to build Unity software for Android is a major benefit. The ability to run Unity software in a web browser was especially popular in the studies. Fredrick [2014], Laksono and Aditya [2019], Rafiee and others [2017], and Wang and others [2012] all report having, in addition to a Windows version, built a version of the software for deployment on the web either using the Unity web player or WebGL. Laksono and Aditya [2019], and Rafiee and others [2017] report having built a mobile version of their software as well.

5.1.4 External assets and plug-ins

The use of external assets, plug-ins and libraries was reported in six studies. However, none of the studies used similar or comparable external assets. This is most likely explained by the major differences in fields of research. This indicates that Unity's asset ecosystem is quite comprehensive.

5.1.5 Scripting

Scripting was a commonly discussed topic in the case studies: the vast majority of papers at least mentioned it. One clear exception is the study by Fredrick [2014], whose project description didn't imply scripting. Some studies explicitly reported use cases for

scripting. For example, Laksono and Aditya [2019] used C# scripts for camera navigation, player movements, as well as user interfaces and their interactions.

The object-oriented nature of C# was commented on in two studies [Zhang *et al.*, 2019; Polcar *et al.*, 2017]. Polcar and others [2017] argue that object-oriented programming helps them make their simulation project agent based and further allows the aim of the simulation to be changed easily.

Nearly all projects used C# for scripting, but some alternatives were mentioned. Buyuksalih and others [2017] imply using UnityScript and mention Boo and C# as alternatives, while Nakano and others [2016] mention just JavaScript as an alternative to C#. UnityScript is derived from JavaScript, and was called JavaScript in the Unity editor [Fine, 2017]. Further, Unity has dropped support for UnityScript and Boo after these studies were published, leaving only C# [Fine, 2017].

5.1.6 VR-capabilities

Three studies that chose Unity used it to develop VR-applications. Nakano and others [2016] chose Unity for it's support for the Oculus interface. Zhang and others [2019] chose Oculus products for their software implementation as well. Ratamero and others [2018] chose Unity because of its support for the HTC Vive headset and for providing standard, easy to use, VR features.

All studies discussed in this section belong to the field of bioinformatics, which indicates that the field is experimenting with new methods of visualisation. Although not quite VR, Wiebrands and others [2018], also a bioinformatics visualisation project, introduced new molecular data visualisation method with the HIVE Cylinder display.

5.2 Challenges

The challenges interpreted from the studies had less common themes. Similarly as in Section 5.1., Table 5 shows the case studies and challenge categories, details of which will be discussed in the following sections.

	File format support	Documentation and support	Performance	Coordinate system
Buyuksalih et al. [2017]	x	x	x	x
Laksono & Aditya [2019]	x		x	
Nakano et al. [2016]	x	(x)		
Polcar et al. [2017]			x	
Ratamero et al. [2018]	x			
Wang et al. [2012]	x			x
Wiebrands et al. [2018]			x	

Table 5: Challenge categories interpreted from case studies. This table can be read the same way as Table 4 in Section 5.1.

5.2.1 File format support

Different file formats can be used to save different kinds of data more efficiently. Same kind of data can have multiple different file formats, such as files describing 3D models. Sometimes software that use same kind of data cannot read the same file formats. Fortunately files can be converted from one format to another. However, this sometimes requires more work and the use of 3rd party software from the user.

Unity's 3D file format support was mentioned in a few case studies. All case studies mentioning file formats reported having to do file conversions to import their 3D files to Unity. Buyuksalih and others [2017] and Laksono & Aditya [2019] report converting GIS format files, such as Shapefile among others, to FBX format to import them to Unity. Buyuksalih and others [2017] report loss of semantic data during the conversion process, while Laksono & Aditya report having done data editing in addition to the conversion process. Nakano and others [2016] report converting Wavefront .obj and .mtl files to FBX files. Ratamero and others [2018] report converting PDB format files to OBJ files to import them to Unity and comment that importing 3D models to Unity while keeping correct colours can be tricky. Wang and others [2012] report using tools

to convert 3D files produced by their tools to FBX format and creating a customized code interface for their use in Unity.

Unity supports model file formats from multiple different software, mainly common modelling programs [Unity 2021d]. The file format conversions discussed before were done on file formats commonly used in specialized applications, such as GIS files and molecular data files. These file formats likely won't see native support in Unity due to their small user base among Unity users.

5.2.2 Documentation and support

Documentation is crucial in software development because it works as the user manual on using a specific technology. Tutorials and sample projects help expand the documentation and make it easier to understand. These factors were discussed in a few case studies.

Buyuksalih and others [2017] state that most of Unity's documentation is applicable only for Unity's designed purpose: game development. Since their case study was on 3D visualization of spatial objects they found that manuals, tutorials and sample projects made for Unity had limited use for their project. Buyuksalih and others consider Unity a very specialized tool for 3D visualisation with various limitations and drawbacks. They argue that this contributes to the limited number of users using Unity for this purpose and thus explains their experience of limited relevant documentation and support.

As virtual reality projects, the case studies by Zhang and others [2019] and Nakano and others [2016] both report a different experience in regards to documentation. Zhang and others [2019] report that they chose Unity over Unreal Engine because Unity has ample online resources. Online resources here most likely refer to the aforementioned documentation and tutorials. Nakano and others [2016] argue that Unity is not designed with a specific game in mind providing them with a "clean slate" onto which to build their simulation. They compare Unity to Unreal Engine, the latter they argue is designed with first-person shooters in mind.

5.2.3 Performance

Some studies did not find Unity's performance adequate. Buyuksalih and others [2017] argued that, at least from a graphical point of view, the Unreal engine performs better than Unity.

Wiebrands and others [2018] reported degrading performance when the number of active objects reached tens of thousands. Unity implemented GPU instancing during the development of the project but this did not solve the performance problems. They had to

make some workarounds to improve the performance but this came at a cost of some visualization quality.

Polcar and others [2017] concluded through testing that simulations running in Unity cannot be sped up enough to be suitable for long term simulation runs. Unity applications can be sped up through scripts. However, this causes the application to demand more computing resources form the host computer.

Laksono and Aditya [2019] reported that the performance of Unity 3D applications in web browsers relies heavily on the internet capabilities and computer hardware of the end-user. Poor internet connection caused significant loading times for the 3D visualisation. The cause of the slow loading time was found to be the 3D model that needed to be downloaded on the end-user's computer. Decimating the mesh in Blender resulted in smaller model file size and faster loading time.

5.2.4 Coordinate system

Coordinate systems are found in all software that handle two or three dimensional data. However, there may be differences in the implementation of the coordinate system between different software, such as naming the coordinate axes.

Wang and others [2012] pointed out that Unity uses a coordinate system where X- and Z-axes form the surface plane with Y-axis pointing up. They state that this is different from the GIS field, where commonly X and Y form the surface plane and Z points up. Buyuksalih and others [2017] reported coordinate system interoperability problems with Unity and found no tools for coordinate conversions or transformations.

6 Conclusions

This systematic literature review explored Unity's usage in visualization, simulation and modelling in research. Unity is a development platform for creating 3D and 2D applications, most notably video games. The literature review included case studies that reported usage of Unity. As shown in this paper, Unity has a wide range of applications in several areas of research such as bioinformatics visualization, architectural modelling and engineering simulations.

This literature review analysed the case studies to find reports on the different advantages and challenges of using Unity. The most commonly reported advantages of Unity were its accessibility, licensing and scripting support. Unity's accessibility refers to the Unity editor's ease of use which allowed for fast prototyping and development. Unity's licensing was deemed suitable for research projects, as it is low cost for research projects and in some cases even free for individuals and educational institutions. Unity's scripting capabilities were used in nearly all of the included case studies.

Unity's support for cross-platform deployment, virtual reality, as well as external assets and plug-ins were also reported as important in some case studies.

The commonly reported challenges of Unity included restricted file format support, lacking documentation for research purposes, and software performance. Unity only supports select file formats, so case studies using different formats had to do file conversions. File conversions involve additional work and may cause loss of information between formats. Unity's documentation and support focused on game development, which wasn't always helpful in research projects. Finally, Unity's performance was deemed lacking in projects where the simulation had to be sped up or needed to render large amounts of or highly detailed objects.

There were limitations to the way this thesis was implemented. The collection of Unity's advantages and disadvantages in research required significant amount of interpretation of qualitative data. This is even more significant when considering that the case studies are very loosely related by topic. This might make it difficult to justify comparison of collected data between different case studies. Fortunately comparisons within the categories discussed in Section 4, especially within the Bioinformatics category, is more feasible. However, the advantages and challenges found in this thesis, when taken into account, could help in assessing usage of Unity for research projects. A better way to collect data on Unity's advantages and challenges could be, for example, surveys, which would provide consistent and highly comparable data. This is something for future studies to address, however.

To conclude, the systematic literature review answers the research question by presenting relevant case studies from multiple research fields, and analysing and reporting their experiences with Unity. The results of this paper may be useful to projects looking for a software based visualisation, simulation or modelling platform. The presented advantages and disadvantages should be considered when choosing a software framework in the project planning phase.

References

Azarakhsh Rafiee, Pim Van der Male, Eduardo Dias, and Henk Scholten. 2017. Developing a wind turbine planning platform: Integration of "sound propagation model–GIS-game engine" triplet. Environ. Model. Softw. 95, (2017), 326–343. DOI:*https://doi.org/10.1016/j.envsoft.2017.06.019*

Bernhard Klein. 2016. Managing the Scalability of Visual Exploration Using Game Engines to Analyse UHI Scenarios. In Procedia Engineering 169, Elsevier Ltd, 272–279. DOI:https://doi.org/10.1016/j.proeng.2016.10.033

C. Masato Nakano, Erick Moen, Hye Suk Byun, Heng Ma, Bradley Newman, Alexander McDowell, Tao Wei, and Mohamed Y. El-Naggar. 2016. IBET: Immersive visualization of biological electron-transfer dynamics. J. Mol. Graph. Model. 65, (April 2016), 94–99. DOI:*https://doi.org/10.1016/j.jmgm.2016.02.009*

Dany Laksono and Trias Aditya. 2019. Utilizing a game engine for interactive 3D topographic data visualization. ISPRS Int. J. Geo-Information 8, 8, Article 361 (August 2019), 18 pages. DOI:*https://doi.org/10.3390/ijgi8080361*

David Fredrick. 2014. Time.deltaTime: the vicissitudes of presence in visualizing Roman houses with game engine technology. AI Soc. 29, 4 (October 2014), 461–472. DOI: *https://doi.org/10.1007/s00146-013-0488-5*

Erick Martins Ratamero, Dom Bellini, Christopher G. Dowson, and Rudolf A. Römer. 2018. Touching proteins with virtual bare hands: Visualizing protein–drug complexes and their dynamics in self-made virtual reality using gaming hardware. J. Comput. Aided. Mol. Des. 32, 6 (June 2018), 703–709. DOI:*https://doi.org/10.1007/s10822-018-0123-0*

I. Buyuksalih, S. Bayburt, G. Buyuksalih, A. P. Baskaraca, H. Karim, and A. A. Rahman. 2017. 3D modelling and visualization based on the Unity game engine – advantages and challenges. In ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Copernicus GmbH, 161–166. DOI:*https://doi.org/10.5194/isprs-annals-IV-4-W4-161-2017*

Jimmy F. Zhang, Alex R. Paciorkowski, Paul A. Craig, and Feng Cui. 2019. BioVR: A platform for virtual reality assisted biological data integration and visualization. BMC Bioinformatics 20, 1 (February 2019), 78–88. DOI:*https://doi.org/10.1186/s12859-019-2666-z*

Jiri Polcar, Petr Horejsi, Pavel Kopecek, and Muhammad Latif. 2017. Using unity3D as an elevator simulation tool. In Annals of DAAAM and Proceedings of the International DAAAM Symposium, Danube Adria Association for Automation and Manufacturing, DAAAM, 517–522. DOI:*https://doi.org/10.2507/28th.daaam.proceedings.073*

Jue Wang, Keith Bennett, and Edward Guinness. 2012. Virtual Astronaut for Scientific Visualization—A Prototype for Santa Maria Crater on Mars. Futur. Internet 4, 4 (December 2012), 1049–1068. DOI:*https://doi.org/10.3390/fi4041049*

Michael Wiebrands, Chris J. Malajczuk, Andrew J. Woods, Andrew L. Rohl, and Ricardo L. Mancera. 2018. Molecular Dynamics Visualization (MDV): Stereoscopic 3D Display of Biomolecular Structure and Interactions Using the Unity Game Engine. J. Integr. Bioinform. 15, 2, Article 20180010 (June 2018), 8 pages. DOI:*https://doi.org/10.1515/jib-2018-0010*

Minh Nguyen, Mohammed Melaisi, Brent Cowan, Alvaro Joffre Uribe Quevedo, and Bill Kapralos. 2017. Low-end haptic devices for knee bone drilling in a serious game. World J. Sci. Technol. Sustain. Dev. 14, 2/3 (April 2017), 241–253. DOI:*https://doi.org/10.1108/wjstsd-07-2016-0047*

Richard Fine. 2017. UnityScript’s long ride off into the sunset. Retrieved from *https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/* on 19.2.2021.

Unity Technologies. 2021a. Editor. Retrieved from *https://unity3d.com/unity/editor* on 14.2.2021.

Unity Technologies. 2021b. Public relations. Retrieved from *https://unity3d.com/public-relations* on 14.2.2021.

Unity Technologies. 2021c. Unity. Retrieved from *http://unity.com* on 14.2.2021.

Unity Technologies. 2021d. Manual. Retrieved from *https://docs.unity3d.com/Manual/index.html* on 14.2.2021.

Unity Technologies. 2021e. Scripting API. Retrieved from *https://docs.unity3d.com/ScriptReference/index.html* on 14.2.2021.

Unity Technologies. 2021f. Compare plans. Retrieved from *https://store.unity.com/compare-plans?currency=USD* on 14.2.2021.

Unity Technologies. 2021g. The Unity License Grant Program. Retrieved from *https://unity.com/education/license-grant-program* on 14.2.2021.

Unity Technologies. 2021h. Frequently asked questions. Retrieved from *https://unity3d.com/unity/faq?_ga=2.5998074.838570682.1611672169-1982909119.1608186040* on 14.2.2021.

Unity Technologies. 2021i. Unity Terms of Service. Retrieved from *https://unity3d.com/legal/terms-of-service* on 14.2.2021.

Unity Technologies. 2021j. Unity Asset Store. Retrieved from *https://assetstore.unity.com/* on 4.7.2021

Unreal engine. 2021. Unreal Engine. Epic games, Inc. Retrieved from *https://www.unrealengine.com/en-US/* on 14.2.2021.