UNITYREV - BRIDGING THE GAP BETWEEN BIM AUTHORING PLATFORMS AND GAME ENGINES BY CREATING A REAL-TIME BI-DIRECTIONAL EXCHANGE OF BIM DATA

KIERAN W. MAY¹, JAMES WALSH², ROSS T. SMITH³, NING GU⁴ and BRUCE H. THOMAS⁵
¹,²,³,⁴,⁵University of South Australia, Australian Research Centre for Interactive and Virtual Environment
¹ kieran.may@mymail.unisa.edu.au , 0000-0002-5461-4927
²,³,⁴,⁵{james.walsh|ross.smith|ning.gu|bruce.thomas}@unisa.edu.au,
² 0000-0002-4822-990X, ³ 0000-0002-9044-9199, ⁴ 0000-0002-5555-9165, ⁵ 0000-0002-9148-085X

Abstract. We present UnityRev: An open-source software package that enables a workflow designed to facilitate a real-time bi-directional and synchronous exchange of Building Information Modelling (BIM) data, by creating a direct link between a BIM authoring platform (i.e. Autodesk Revit) and a game engine (i.e. Unity 3D). Although previous works have explored the integration of BIM with game engines, the currently available tools are limited to a non-synchronous or unidirectional exchange of BIM data, and they do not address specific design issues required to make a BIM authoring platform and game engine compatible (i.e. parametric modelling). This paper describes our software which consists of a compact overview of the system, including design decisions, implementation details, and system capabilities. Two example applications are presented as concept demonstrators to showcase practical collaborative use-case scenarios between BIM authoring platforms and game engines which were not previously achievable without a real-time bi-directional workflow. This work will expand future Computer Aided Architectural Design (CAAD) research, and more specifically, Virtual Reality (VR)/Augmented Reality (AR) based BIM development and integration, by providing new possibilities and bridging the gap between BIM authoring platforms and game engines. The application of the system as demonstrated in the paper for real-time lighting performance simulation contributes to achieving the UN Sustainable Development Goal 11: Sustainable Cities and Communities.

Keywords. Building Information Modelling; Game Engines; Revit; Unity; Virtual Reality; Augmented Reality; Lighting Performance Simulation; SDG 11.
1. Introduction

This paper presents UnityRev, an open-source and freely available software package that enables a workflow designed to facilitate a real-time bi-directional and synchronous exchange of Building Information Modelling (BIM) data between a BIM authoring platform (i.e. Autodesk Revit) and a game engine (i.e. Unity 3D). Game engines play a pivotal role in recent research, as they act as a stepping-stone for researchers to virtually prototype and build software applications by providing them with a set of software development tools. Previous works comparing game engines and BIM authoring platforms have highlighted three primary motivations for using game engines for architectural representation. Specifically, the interaction, visualisation, and real-time capabilities associated with game engines provide a layer of functionality that BIM authoring platforms are unable to provide (Pelosi, 2010, Boeykens, 2011). Over the last decade, the usage of game engines within Computer Aided Architectural Design (CAAD) research has risen significantly with a large focus on incorporating BIM models with advanced visualisation tools such as Virtual Reality (VR) and Augmented Reality (AR) displays. However, despite the importance of game engines in current VR/AR BIM research, the ability to communicate BIM data between BIM authoring platforms has remained a consistent issue for researchers working within this particular field.

To bridge the gap between BIM authoring platforms and game engines, various tools have been released to support the integration of BIM within game engines; however, these tools lack the capability to support a natural real-time, and synchronous exchange without loss of BIM data. Due to the uni-directional flow of data from these tools, manipulations that occur to the BIM model within game engines are not synchronised back to the BIM authoring platform that contains the original BIM model. Due to the limited tools available, and lack of BIM support provided by game engines, many researchers utilising game engines to prototype BIM-based applications export BIM models from BIM authoring platforms into a file format that is natively supported by the game engine. This process generally involves exporting a BIM model from Revit as a Filmbox (FBX), Object (OBJ) file or other similar data formats supported by game engines that only provide geometric or partial metadata (Bille et al., 2014). As a result, a large portion of the metadata (e.g. cost, schedule, manufacturer, etc.) associated with the geometric model is lost, subsequently transforming the information-rich BIM model into a generic three-dimensional Computer-Aided Design (CAD) model. We believe this limitation to be one of the major gaps in the adoption of immersive BIM-based research within the Architectural, Engineering, and Construction (AEC) industrial sectors. Recent works from the authors have a validated this claim where a workshop was held by the researchers with four AEC industry representatives. The AEC industry representatives stated that the loss of BIM data caused by cross-platform BIM communication was one of the major barriers associated with BIMs, and the integration of VR, and AR applications with BIM models (May et al., 2022).

To address the aforementioned issues, an open-source workflow capable of facilitating a real-time bi-directional and synchronous exchange of BIM data between BIM authoring platforms and game engines could be particularly useful to addressing one of the major research gaps associated with the process of integrating BIM with
immersive VR/AR displays and game engines.

The specific contributions of this paper are:

1. An open-source software package that enables a workflow for linking Unity and Revit through a bi-directional synchronous exchange of BIM data.
2. An approach for enabling parametric CAAD modelling within a physics-driven game engine.
3. Two example applications highlighting the potential advantages of the proposed workflow and demonstrating two scenarios that were not previously possible without the use of a synchronous bi-directional BIM workflow.

In the remainder of this paper, we explore previous works in the field, and identify the common gaps and limitations of currently available tools and literature. We then present the system capabilities of the 'UnityRev' software package that enables the workflow to achieve a bi-directional synchronisation of BIM data. Two example applications are presented as concept demonstrators to showcase practical collaborative use-case scenarios linking BIM authoring platforms and game engines using UnityRev. The application of the system as demonstrated in the following sections for a VR-based Lighting Performance Simulator addresses the UN Sustainable Development Goal 11: Sustainable Cities and Communities. We released UnityRev as an open-source GitHub repository under the MIT license (https://github.com/kieran196/UnityRev-ProjectExchange).

2. Related Works

To identify common tools and previous research exploring the process of linking BIM authoring platforms and game engines, we conducted a literature review focusing specifically on asynchronous and synchronous workflows designed for CAD/BIM authoring platforms to game engine integration.

2.1. ASYNCHRONOUS WORKFLOWS

The most widely adopted approach for the transferal of BIM data across BIM authoring platforms is using the Industry Foundation Class (IFC). IFC is a standardised BIM file-format supported by several BIM-driven CAD platforms. The IFC data format supports the transferal of BIM data by storing the metadata, and geometric properties associated with a BIM into a single file format. Although IFC can maintain partial metadata properties associated with a BIM model, modern game engines such as Unreal and Unity do not provide native support for IFC. As a result, third-party libraries have been released to support parsing IFC files into game engines. Additionally, researchers have developed their own systems capable of parsing IFC files into game engines to provide partial uni-directional integration of BIM data. Previous researchers who utilised a Revit to Unity workflow reported limitations associated with using an IFC-based workflow due to IFC standard only providing partial metadata (Motamedi et al., 2017).

Due to the complex structure of the IFC data model, difficulties emerge when attempting to write BIM changes directly back to the original IFC file, making it significantly challenging to achieve a two-way transferal of BIM data when strictly using IFC. However, (Nandavar et al., 2018) were successful in achieving this by proposing an XML-based approach to store manipulations that occurred to the BIM
model within VR (i.e. Unity). The XML file could then be parsed back to the original IFC file using an open-source IFC management software: xBIM toolkit (Lockley et al., 2017). Similar investigations have also been conducted by (Khalili, 2021); however, using the FBX data model which is natively supported by game engines as opposed to IFC. Our review shows that, the primary advantage of an IFC-based approach is the interoperability and adaptability of the workflow, as it can be integrated with any BIM authoring platform which provides support for IFC. This is a considerable step towards achieving an intuitive integration of BIM data across all BIM authoring platforms and game engines.

2.2. SYNCHRONOUS WORKFLOWS

Fewer works have demonstrated the ability to achieve a real-time exchange of BIM data between BIM authoring platforms and game engines. (Du et al., 2018) created a system called 'BIM-VR Real-time Synchronization' (BVRS), which can perform a real-time synchronous uni-directional exchange of BIM data by autonomously synchronising changes made to the BIM model in Revit back to Unity in real-time. This was achieved by creating a Revit plugin that constantly sends manipulations made to the BIM model within Revit to a cloud-based server that stores and transfers data based on IFC.

In 2018, (Nandavar et al., 2018) identified Fuzor and BIMXplorer as the only two publicly available software packages capable of achieving a complete exchange of BIM data between BIM and VR. Earlier works by (Chotrov and Maleshkov, 2013) also presented an approach using a TCP/IP network communication that achieved a bi-directional synchronisation of CAD geometry between a CAD platform (Solid-works) and VR application. However, all three described systems utilised a customised VR rendering pipeline to achieve this communication as opposed to game engines.

The Unity development team recently released Unity Reflect (Reflect, 2021), which we believe to date is the most advanced publicly available software package capable of achieving a linkage between a BIM authoring platform (Revit) and game engine (Unity). Unity Reflect provides a uni-directional exchange of BIM data by instantaneously sending data from Revit to Unity across a server. The primary limitation we have identified with Unity Reflect is that the transferal of data flows in one direction, and as a result, manipulations that occur to the BIM model within Unity are not sent back to the original BIM model. Furthermore, modifications made to the BIM model in Revit are not updated within Unity in real-time, and users are required to reload the model to synchronise changes from Revit. Both limitations are specifically addressed in UnityRev.

(Edwards et al., 2015) demonstrated the ability to incorporate a network to bi-directionally exchange BIM data between a BIM authoring (Revit) and game engine (Unity). A custom OBJ data-model was used to facilitate the initial geometry associated with the BIM model into Unity. The authors, however, reported they encountered various issues such as manipulations made to the position of elements within VR not updating correctly within Revit.

Recently, (Harlan et al., 2020) presented a three-module system to integrate a CAD platform (Simens NX) and game engine (Unreal). The system presented a unique
approach capable of loading and reloading meshes at run-time by accessing CAD functionality through the NXOpen C++ API. The authors are actively using their proposed software to research hand-tracking based natural user-interfaces for design sketch creation in VR.

2.3. SUMMARY OF PREVIOUS WORKS

In summary, UnityRev addresses the following issues and limitations associated with the prior works discussed above and the currently available tools that produce a link between a BIM authoring platform and game engine:

Of the available CAD to game engine workflows: No tools provide a bi-directional exchange of BIM data and only supports a CAD to game engine directional flow of data. No tools support a real-time synchronous transferal of BIM data. Only one tool (Unity Reflect) can achieve a complete integration of BIM metadata into a game engine with constraints as discussed above.

Of the available CAD/BIM authoring platform to game engine workflows from our review: No prior works produced a workflow capable of supporting a game engine to adhere to the parametric CAAD modelling platforms. No prior works produced a workflow capable of supporting a real-time two-way transferal of BIM data with geometric orientation and geometric mesh manipulation.

3. UnityRev: System Overview

To enable the real-time integration between BIM and game engines, we developed a customised network based on a Transmission Control Protocol and Internet Protocol (TCP/IP) network architecture to facilitate the exchange of BIM data between Revit, a BIM authoring platform and Unity, a game engine. A local server is set up, which is hosted from the Unity software, with Revit connecting to the server as a client. To achieve this connection from the BIM authoring platform, a plugin was developed based on the Revit API, which is run as an add-in directly from Revit. The process of transmitting data between the server and client consists of retrieving the element IDs, position, rotation, scale, metadata, and mesh data via the Revit API and converting it into a bytes array, which is then sent through a socket. This data is then decoded and parsed by the receiver to synchronise modifications within the corresponding software.

A custom network protocol was developed to exchange commands between the server and client to represent the transmission of specific data. Similarly, BIM elements are identified when being transferred across platforms based on their unique IDs, which are automatically generated within Revit. An error-checking system was integrated into the system, ensuring that only valid actions adhering to Revit's parametric modelling can be supported within the Unity engine. The software was written entirely in the C# programming language.

3.1. ADHERING TO PARAMETRIC CAAD MODELLING

Typical BIM authoring platforms such as Autodesk Revit provide parametric modelling to assist users in correctly designing architectural structures. However, physics-based game engines such as Unity and Unreal were not designed to support BIM or CAAD and therefore do not follow parametric modelling rules and
conventions. For example, if a user manipulated the position of a door, causing it to become unattached or floating in a scene within a game engine, this would be perfectly valid manipulation. However, within a parametric based BIM authoring platform, the same manipulation would violate the built-in parametric constraints, and result in an exception to occur. As a result, ensuring that Unity adheres to the specific parametric rules of Revit is a fundamental step in ensuring the compatibility of the BIM authoring platform and game engine when linking the two. However, due to the complex nature of parametric modelling, it was not realistic to re-write the parametric constraints within Unity from scratch. Therefore, our solution to solve this problem is to leverage the parametric functionality in Revit by querying the Revit API whenever a manipulation occurs to an element's geometry within a game engine. For example, if the position of an element is modified in Unity, the new coordinate values are sent to the Revit client, where an error-check is performed to determine whether the new coordinates of an element are valid based on the parametric constraints of Revit. If the new position is invalid, an exception is thrown, and the Revit client sends a query back to the Unity server, and the element within the Unity scene is automatically reverted to its previous position.

3.2. REAL-TIME RETRIEVAL AND MANIPULATION OF BIM DATA

The ability to seamlessly transfer BIM metadata between a BIM authoring platform and game engine in real-time with no loss of data is one of the significant features provided by UnityRev. Upon running the Revit plugin, the Revit client will connect to the server and instantaneously send all parametric metadata associated with each element to the Unity server. The metadata is then retrieved in Unity and stored in a 'RevAttributes' class, which is attached as a component to each BIM element within the Unity scene. Within this class, the parametric metadata associated with each element is stored in a dictionary data structure where the Key is the unique name of each parameter, and the value is the value of each parameter. Real-time manipulation of BIM metadata is handled by having the Revit and Unity software continuously checking whether the value of element parameters have been modified. Once a modification occurs to an element's parameter value, the new parameter value is sent back to the corresponding software and updated. Whilst the metadata is present within the Unity inspector, it is not directly utilised within the scene. However, users can easily extract and use the necessary metadata within their Unity applications. This is demonstrated in Section 4.2 where lighting metadata is extracted from a Revit lighting fixture and used directly within the scene by a Unity-based lighting performance simulator.

3.3. GEOMETRIC DATA MANIPULATION AND SCALING

Initially, we encountered several issues associated with sending the direct coordinates of elements from Unity to Revit. Therefore, we employed an approach that sends an offset Vector representing the coordinate change between the previous and current position of elements over a one-second interval. The purpose of this one-second delay is to ensure the server is not overloaded by requests due to the Unity engine running at multiple frames per second. This delay is not required in Revit due to the non-real-time
structure of Revit when modifying the position, rotation, or scale of an element.

Current game engines typically utilise a three-axis vector-based scaling system wherein a scale of one is equal to the original scale of the geometric model. In contrast, Revit splits the scaling of elements into two separate three-axis vectors: BoundingBoxMin, and BoundingBoxMax. The scale is represented based on the constraints of an element using an imperial-based measurement system. To convert this into a scaling-based system supported by Unity, we rewrote a customised scaling system in Revit to adhere to the scaling requirements of Unity. Additionally, this approach allowed us to bypass the need to account for different measurement systems when sending scaling data between the two platforms.

4. Demonstration of UnityRev's Capabilities

The real-time and synchronous capabilities of UnityRev present new opportunities for collaboration between a BIM authoring platform and game engine, which were not previously possible. In this section, we present two example applications that use UnityRev as a workflow to demonstrate how it could be incorporated within practical use-case scenarios. The first example application uses UnityRev to create a collaborative design platform between a Revit (CAAD) user and a Unity (VR) user. The second example application connects the Revit software with a real-time lighting performance simulator developed in the Unity game engine.

4.1. COLLABORATIVE ARCHITECTURAL DESIGN BETWEEN BIM AUTHORING PLATFORMS AND GAME ENGINES

The first application demonstrates the ability to create a collaborative architectural design scenario between a Revit and Unity user. Using the Unity game engine, an immersive VR application was developed, allowing users to manipulate the geometry (e.g. position, rotation, and scale) of elements within a virtual environment (VE). The following two three-dimensional interaction techniques supporting six degrees of freedom (DOF) were implemented using the 3D User Interaction Toolkit (May et al., 2019): 3D Bubble Cursor (Vanacken et al., 2007) and Flashlight (Liang and Green, 1994).

The motivation behind the presented example application is to explore and encourage the development of alternative CAD tools in architectural design and collaboration. In this case, we demonstrate a scenario where a VR user using alternative 6DOF 3D interaction techniques can manipulate elements whilst simultaneously collaborating with a desktop-based user using a standard 2DOF mouse input. Although we believe a cursor is overall more intuitive, and suited for the complex user interface, and fine movements required in a CAD environment, limited research has been conducted exploring the potential benefits of using immersive 6DOF interaction in CAD. We believe one aspect linked to the limited research in this field could be due to the difficulties associated with integrating sophisticated parametric CAAD modelling within a game engine making it difficult to realistically compare a CAD platform with a game engine-based VR application. However, using UnityRev enables this link to be achieved by connecting the two platforms and providing a layer of support for parametric modelling within a game engine.
4.2. LINKING A REAL-TIME LIGHTING PERFORMANCE SIMULATOR WITH A BIM AUTHORING PLATFORM FOR COLLABORATIVE DESIGN

The second application integrates a VR-based Lighting Performance Simulator developed by the authors (May et al., 2020) using Unity to simulate the lighting performance of the environment through BIM. The lighting metadata (lumens, wattage, and range) is retrieved from the BIM model's lighting fixture elements to initialise the simulated artificial lighting within the game engine. The lux and unified glare rating (UGR) values outputted from the simulator are sent and linked back to the Revit model in real-time using UnityRev.

The presented example application not only provides an approach for integrating a game-engine based simulator directly with Revit, but also autonomously synchronising lighting performance data back to the Revit model, thus maintaining and updating the original BIM model. Additionally, typically architects would have to complete a draft of their architectural design prior to using a cross-platform simulator. However, using UnityRev provides the ability for users to run simulations much earlier and more
readily within the design phase. This could potentially allow an architect to identify design issues and discrepancies much earlier in the design and mitigate the need for major re(iterations) later, resulting in time and cost saving.

5. Limitations

The initial development of UnityRev could be potentially improved in various aspects. Firstly, the initial implementation of the network is not well optimised and is designed to focus primarily on the functionality goals we set out to achieve as opposed to performance. Additionally, limited user-testing has been carried out on the initial software, and comprehensive user-testing to identify issues could significantly improve the usability of the software. We also encountered limitations associated with using the Revit API; for example, creating new elements is only capable of flowing from a Revit to Unity direction. Lastly, the dependency of the Revit API to achieve this real-time two-way communication between a BIM authoring platform and game engine prevents us from providing future support for UnityRev on alternative BIM authoring platforms. These limitations are being addressed in the current enhancement of the research.

6. Conclusion

This paper presented UnityRev: An open-sourced software package that enables a network-based workflow designed to bridge the gap between a BIM authoring platform (Revit) and game engine (Unity) by creating a real-time bi-directional exchange of BIM data. In summary, we conducted a literature review in the field and an analysis of previous CAD/BIM authoring platforms to game engine workflows and their implementations. We then presented a system overview of UnityRev which describes the system capabilities and how UnityRev addresses specific gaps identified in the review and analysis. Next, we presented two example applications as concept demonstrators that aimed to showcase two practical collaborative use-case scenarios between a BIM authoring platform and game engine that were not previously achievable without the use of the synchronous bi-directional workflow being proposed. The general future directions of this work can be improving upon the software by addressing the limitations described in Section 5. We are also motivated by the recent announcements from Facebook and Microsoft that are working towards creating a metaverse - a network of 3D virtual worlds. We envisage the tools presented in this paper can provide a bridge between BIM allowing bi-directional communications to extend the future capabilities of the metaverse. This work was specifically designed to pilot for a larger project by the authors, which focuses on integrating immersive displays with building performance simulators for real-time visualisation and simulation of architectural design. Our broader motivation of this research is to advance BIM research by encouraging new possibilities and use-cases that can be achieved through VR/AR-based approaches adopting UnityRev. As described above, we demonstrated in Section 4.2 that a game-engine based VR building performance simulator could be efficiently integrated with BIM models to support the architectural design process. This highlights a scenario where UnityRev could be used in green building research, potentially leading to designing more energy-efficient and sustainable buildings and cities.
References


