Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Interactive CFD simulations with virtual reality to support learning in mixing

Serkan Solmaz and Tom Van Gerven*
* Corresponding author, tom.vangerven@kuleuven.be
Department of Chemical Engineering, KU Leuven, Celestijnenlaan 200F, B-3001 Leuven, Belgium

Abstract

This study presents a user-friendly digital learning environment with interactive computational fluid dynamics simulations for higher education in chemical engineering. A client-server network is introduced to maintain an automated two-way connection between multiple CFD solvers and a game engine as a cross-platform development tool. A modular software development approach is pursued to obtain long-lasting connected digital applications. To prevail over heavy calculations and standalone features, a content delivery network is structured, through which any end-user devices are made employable regardless of hardware performance. From desktop to virtual reality applications, multiple digital platforms are demonstrated within a case study of mixing. The system performance of technical integrity is assessed to ensure a feasible and effective connection through the client-server network. A task-centered educational model is implemented to design digital learning environments concerning complex skills to be developed by learners. Discussions are drawn upon the applicability of the system and also integrity to traditional teaching practices.

Keywords: digitalization, higher education, mixing, computational fluid dynamics, virtual reality, automation

1. Introduction

Digital devices are becoming an intimate part of learning since the turn of the century. European Union has declared several research plans to adopt digital resources in education and training (European Commission, 2011), including a recently debuted Digital Education Action Plan for 2021-2027 (European Commission, 2020). The plans have been proposing concrete steps to lead to a high-performing digital education ecosystem with digitally competent human resources. Principally, high-quality learning content and user-friendly tools have been strictly underlined together with the emergency of augmented and virtual reality (AR/VR) technologies.

Chemical engineering content is ever-evolving to find efficient processes. Notably, process intensification has been receiving much attention in recent years from academicians and industrialists (Fernandez Rivas et al., 2020a, 2020b). Mixing is one of the core unit operations in chemical engineering taking part in almost every chemical process. For instance, in order to develop a millifluidic crystallizer to produce pharmaceuticals within an efficient process, an engineering student should be able to comprehend and interpret transport phenomena, multiscale and multiphysics interactions. Not only handbook solutions and experimental studies, but also multiphysics computational fluid dynamics (CFD) simulations can be blended in the technical work to make progress in an efficient manner (Ge et al., 2019).

CFD simulations are essential tools to support engineering design and analysis. Cost time benefits, advanced data exploration and fast prototyping are the main advantages of CFD simulations. A deep

understanding of physical, mathematical and computer science is critical to obtain optimal simulation workflows. These require physical accuracy, optimal calculation time and meaningful result extraction. CFD simulations have a dynamic role to play in engineering education, by producing abstract learner content, and facilitating active learning environments (Tian, 2017). However, neither lecturers nor students might be acquiring relevant skills to deal with the complex nature of simulation environments. CFD is generally assumed not to be user-friendly and does not provide supporting content in the context of instructional design. Engineering students can face a high entry barrier in simulation environments developed through conventional workflows.

Game engines have recently shown great potential to develop functional and high-purpose digital environments from mobile to AR/VR applications. In essence, they can effectively serve in the development of head-mounted display (HMD) VR environments where the learning experience is assisted and assessed with simulations and supporting information. These applications can also amplify cognitive skills with advanced human-computer interactions (HCI); while leveraging motivation within user-friendly, fun and engaging environments. A recent study reported that process simulations in a VR environment increase students' proficiency in chemical engineering content (Pirola et al., 2020). Integration of CFD simulation data with game engines has already been demonstrating applications including positively responded user assessments (Huang et al., 2017; Konrad E. R. and Behr, 2020; Lin et al., 2019; Logg et al., 2020; Shi et al., 2020; Wehinger and Flaischlen, 2020). Despite this, user applications are still lacking functional, well-performed and easy-to-implement design elements in the development workflow. Several key challenges are present to adopt AR/VR with technical content in engineering and education (European Commission, 2017; Li et al., 2017; Su et al., 2020). Digital applications are generally hardcoded, platform-specific and standalone; comprising a static CFD post-processing without any connection to a CFD solver or data processor (Li et al., 2017). Technical integration of simulation data with cross-platform software should still be studied to obtain long-lasting digital applications concerning software development workflow, remote connection, data visualization, and user assessment (Su et al., 2020).

Taking on these challenges, this work develops an automated two-way connection between CFD solvers and Unity game engine. In this study, the term two-way coupling refers strictly to a dynamic connection between CFD solvers and user applications built in Unity. The ultimate goal is to develop CFD simulation-assisted user-friendly digital applications to teach mixing in chemical engineering. Whilst the system ensures content-wise accurate and rich environments, it can support life-long learning, enabling simulation-driven learning cycles as supplementary tools to traditional applications. The present study demonstrates significant contributions:

An automated two-way connection between CFD solvers and Unity to perform interactive simulations. A remote content delivery approach is presented with a client-server network model. The integration is strictly tied to modular, automated, easy-to-update, lightweight, end-to-end and open-source utilities. Both OpenFOAM and COMSOL are simultaneously and successfully connected to Unity. Seamless integration is prompted between CFD software and user applications. Automated workflows are extremely flexible and easily be customized for any compelling needs with the modular architecture.

Well-maintained features to interact with CFD simulations in VR. User interaction and system performance are ensured to obtain a feasible development workflow concerning data processing, dataflow, data management and delivery systems in the network. The system also highlights practices to

develop co-simulation and digital twin environments with interactive CFD simulations remotely accessible.

A digital authoring tool to customize the learning environment with CFD simulation data. Thanks to the game engine, the digital environment can be quickly customized upon developers' needs on digital learning content, as well as user interaction and device. A task-centered educational model is implemented to design a digital learning environment based on complex skills to be developed by learners. A VR application is configured to learn macroscale mixing through the analysis of a stirred tank reactor with CFD simulations.

1.1. Background and review

Early research in this field primarily focused on the automation of CFD solvers to produce data for cross-platforms, rather than two-way coupling. Challenges related to a fully coupled computing system stressed the need for an expert to maintain automated data generation and simulation routines (Ham and Golparvar-Fard, 2013). Recently published works mostly studied remote connections to automate CFD solvers to produce data for cross-platforms. Studies reported the importance of pre-optimized models in CFD workflows with constrained parameters that non-expert users can easily interact with (Fukuda et al., 2019; Zhu et al., 2019). Several remote connection models were also exhibited to maintain semi-automated routines for CFD solvers and data processing (He et al., 2019; Kim et al., 2018, 2020; Li et al., 2018), including peer interaction (Li et al., 2018). These studies tended to focus on the domain- and platform-specific applications with hardcoded elements in their systems, thereby posing a danger to be outdated in a short period of time. Above all, manual expert interferences were still required in most cases of two-way connections.

A review of the literature on this matter reported that technical works generally lack automation of dataflow between CFD solvers and user applications. The study further stated that digital environments were built upon very particular data processing methodologies that make any post-production unfeasible due to hardcoded utilities (Li et al., 2017). Recently, a one-way automated connection was proposed to integrate CFD simulation data with game engines (Solmaz and Van Gerven, 2021). Even though a methodology was presented towards a two-way connection, the study mostly investigated robust data processing work rather than connectivity.

Technical works related to fully automated two-way couplings between CFD solvers and cross-platforms have been gaining momentum to develop digital tools for non-expert users. A study proposed a well-maintained network to perform interactive CFD simulations with virtual reality. User interaction with simulation data and dynamic post-processing were suggested through very plausible approaches to bring case-specific but easy-to-customize features (Shi et al., 2020). One of the major drawbacks to adopting this system is the CFD solver in the connection maintained. A Lattice-Boltzmann solver was implemented to run domain-specific CFD simulations for cardiovascular science, which is based on a specific software architecture different than the ones used in CFD solvers such as OpenFOAM, COMSOL, Ansys and STAR-CCM+. Another study took on the difficult task of the automation of the whole process in a digital twin model. It proposed an automated connection utilizing a CFD software architecture that is similar to commonly used ones in academia and industry (Lydon et al., 2019). Particularly interesting was the way in which the study additionally detailed the automation in-depth with a dataflow presented. A user application was presented targeting non-experts to perform CFD simulations with constrained input

parameters. However, the automated routine was built towards a digital twin approach, and the system did not consider either immersive technologies or game engines. There is still considerable ambiguity to maintain an automated two-way connection between commonly used CFD solvers and Unity.

User interaction with simulation parameters was discussed by a few scholars. In general, parameter-based interaction was preferred to acquire user input (Huang et al., 2015; Lydon et al., 2019; Shi et al., 2020). In some cases, manipulation of model geometry with different HCI interfaces was even taken into account implementing hand controllers, haptic devices and image processing. Studies recommended considering HCI interfaces with caution concerning accuracy and steadiness of geometric manipulations made. Limiting user interaction and adding supporting features were found to be an optimal approach to automate systems such as local grid refinement and pre-modeled geometric extrusion. Expert support was strictly mentioned in the development of automated systems to overcoming technical issues that might affect CFD simulation workflows (Lydon et al., 2019; Shi et al., 2020).

Dynamic post-processing of simulation data in the user application was investigated to facilitate a constrained but flexible data exploration tool for non-expert users such as slicing, clipping and animating (Huang et al., 2015; Kim et al., 2018; Shi et al., 2020; Wang et al., 2020). While the authors' position is that the connection is feasible, there are many weaknesses in this concept of which a recent criticism was the following (Su et al., 2020). Either a game engine or user application is employed to process CFD data which is often computationally demanding for end-user devices. Relevant post-processing algorithms to process CFD datasets should be re-coded in the game engine, mainly from the ground up.

Furthermore, cross-platform environments with real-time CFD models have become common in academia (Harwood, 2019; Logg et al., 2020). However, the current state of the research only promotes either very simple or low-fidelity simulations. These would only seem to be helpful in teaching the basics of fluid mechanics.

Lastly, the use of a functional mock-up interface (FMI) is a standard approach becoming popular in the context of interoperability. It is fundamentally a communication hub to manage connections with a common application programming interface (API). The FMI standard is an open-source format enabling co-simulation and model exchange features among heterogeneous platforms. A software should give support on the FMI standard to be tied in a connection pipeline. No study has been found in the literature developing a two-way connection with FMI within the scope of the present work. COMSOL (Dahash et al., 2019) and OpenFOAM (Tian et al., 2018) do not directly give support on the FMI standard yet. In our proposed approach any software with any type of API can be integrated with API-based modular connections.

2. Methodology

2.1. System design: the client-server network

The present study proposes a client-server network to maintain an automated two-way coupling between CFD solvers and user applications. This approach aims to develop user-friendly learning environments with technological advancements. A client-server network is a distributed system in which servers and clients are given specific tasks to maintain a remote connection. Once the client initiates a request; the server receives the request, executes the necessary action, and then responds back to the client. Multiple clients and servers can be implemented to the same network in case the system requires heterogeneous utilities,

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

for example, high-performance computing (HPC) facilities for computationally demanding tasks. The client-server network typically adheres to an internet protocol suite (TCP/IP) with connected utilities.

Fig. 1 presents a client-server network to establish a two-way connection between simulation software and game engine. To achieve this, we propose two subsequent workflows; design and automation. The network provides a flexible environment in which multiple users connect and interact simultaneously. All heavy-duty processing works are executed on the server-side. The client hardware, which employs the user application, is merely utilized to stream the content delivered through the network. The entire network is maintained by the following five executive modules; user application, computational fluid dynamics simulator, simulation datasets, data processing software and game engine. Datasets and data processing software modules are adopted from our previous work to integrate CFD data with Unity (Solmaz and Van Gerven, 2021).

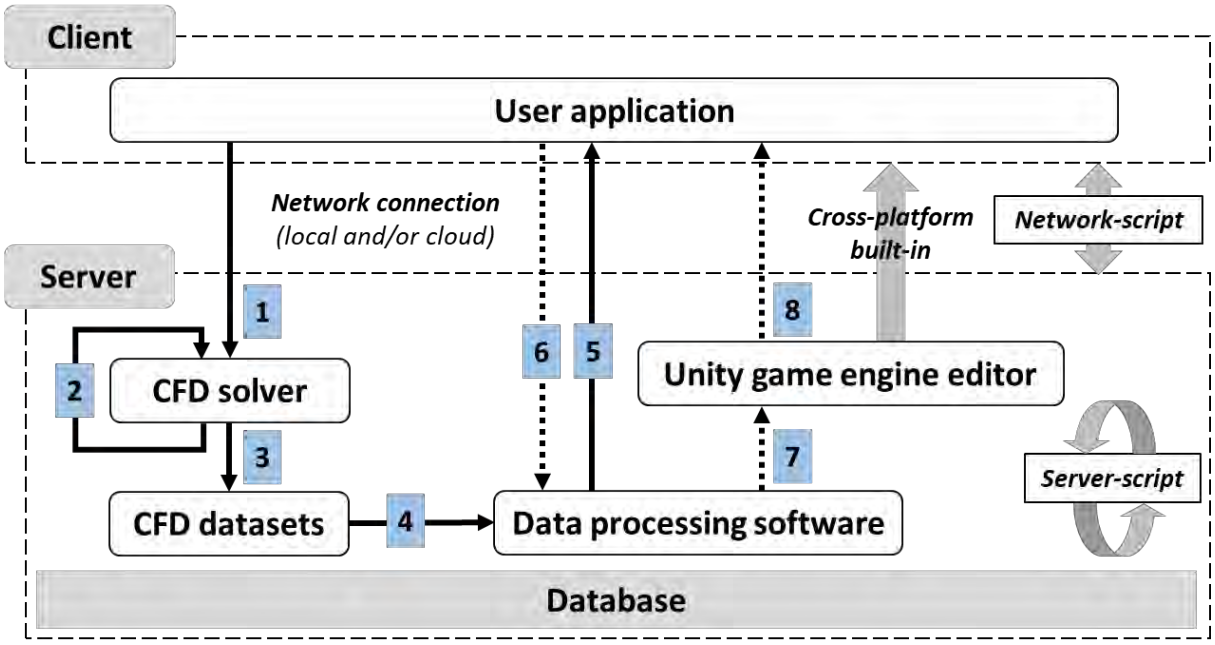


Fig. 1 The client-server network to maintain an automated two-way coupling between CFD solvers and user applications developed with Unity. Connections with dashed lines are optional dataflow pipelines that might be entailed for advanced interactions.

The network consists of one built-in and eight inter-connections as numbered in Fig. 1, whereby in-house Python scripts automatically handle actions inside the network. The server-side is managed via “server-script” to maintain communication among the modules. In comparison, the client-side does not require any script as all essential settings are fixed within the cross-platform built-in. Interaction between the client and server is handled with “network-script”. These in-house scripts are mainly composed of file handling operations with standard Python libraries.

The design workflow is the phase in which a user application is built in Unity only one time. Decisions are made in this phase to draft a baseline connection (from 1 to 5) with intended optional features (from 6 to 8). Updates in user applications proceed with a content delivery network (CDN) created in Unity and then embedded in the client-server architecture. This avoids the requirement for the system to rebuild a new

user application each time if the digital content is updated. Any content in the user environment can be remotely deployed as long as being linked to the CDN.

The automation workflow comprises fully integrated modules. It mainly includes all inter-connections without a built-in. First, the user input in the client is processed in a case file, which is sent to the relevant CFD simulation tool. This case file is then automatically updated in the relevant directory to run a new simulation. Results are saved in native formats and processed further to integrate CFD data into Unity within third and fourth connections. Finally, the data in the user application is updated through the fifth connection. This way, a baseline automated two-way connection is achieved between CFD solvers and user applications.

Furthermore, the client can dynamically post-process simulation results through a restricted well-maintained connection between the user application and data processing software, labeled as the sixth connection. Likewise, a CFD solver is generally composed of a set of manual steps that must be accurately handled by an expert. Complete automation might be challenging due to the software architecture and data encryption in CFD solvers. Hence, an internal connection is separately demonstrated to CFD solvers, to stress the significance of automation using the second connection. In addition, in case the game engine is applied as an intermediate data processor, the seventh and eighth connections can optionally be established without built-in. For instance, to efficiently manage extract-based CFD data in user applications, a pre-processing via addressable asset manager in Unity might be tailored in the workflow.

The network is aimed at heterogeneous and distributed computing facilities, such as cloud and HPC servers. A database management system is prescribed to intelligently orchestrate data-in-use. All computationally intensive and complex data processing tasks are executed in the server. The data in the user application is kept in either the client hardware or database in the server. Each module in the server is inherently connected to the database at the back end.

2.2. Design of the learning environment

Performing CFD simulations requires complex skills to solve engineering problems. Learning in conventional simulation environments often happens by learning-by-doing. The environment does not comprise any assistance except help options relevant to the usability of the tool. This methodology has been heavily criticized by educational scientists and found less efficient than that of traditional instructional designs (Kirschner et al., 2006). Meanwhile, game engines are versatile tools to develop a digital environment in varying contexts. Any assistance can be effortlessly delivered in runtime through guided learning environments.

Several interesting models to design digital learning environments have been researched by educational scientists over the last two decades. The four-component instructional design (4C/ID) model has gained much attention to support complex learning environments in numerous disciplines (Van Merriënboer and Kester, 2014). The model promotes four major components to enable complex learning; learning task, supportive information, just-in-time information and part-task practice. The learning environment in this study (Table 1) is originated from the 4C/ID model utilizing suitable design principles elaborated in the relevant research (Van Merriënboer and Kester, 2014).

In particular, the learning task is organized toward the sequential principle to progressively increase complexity. A tuning approach is pursued to reduce the level of assistance given with supportive and just-in-time information throughout tasks. A smooth entrance, with a tutorial-like simplified very first level, is provided to let students explore the learning environment and usability in the first place. The complexity is then leveled up by tuning working memory and cognitive loads upon components of the model. The part-task practice is considered to improve performance on routine aspects of learning tasks in the simulation workflow. Besides, we also try adapting the following approaches to use multimedia effectively in the learning environment. The multimedia principle is mainly applied prior to the VR experience to give supportive information through pictures and animations with explanatory texts. The signaling principle is implemented for just-in-time information, in which instructions are highlighted in a step-by-step fashion upon the relevant step in the learning task. No study has been found in the literature implementing a task-centered complex learning method with CFD simulations in a digital environment.

Table 1 Implementation of 4C/ID with CFD simulations; an example from a specific learning task.

Component	Principle	Generic content	Example
Learning task	Sequential	Intensification of mixing process in a stirred tank reactor	Assess different rotating impellers in a mixing process
Supportive information	Multimedia	Explains how to interpret simulation data of a specific task	Visual and written information about power draw and shear force
Just-in-time information	Signaling	Step-by-step, timely instructions to present procedure	Highlighting a component in the model
Part-task practice	Recognize-edit-produce	Improve performance on routine aspects with additional practices	Check mesh quality after geometric manipulations

All in all, it is likely that the 4C/ID model can help students to comprehend and operate simulation-driven learning environments easier than learning-by-doing. Students may develop necessary proficiencies as expected from engineers who deal with CFD simulations; remember, understand, apply, analyze, evaluate and create in a limited period of time (Duque et al., 2016; Tian, 2017).

3. Implementation

This section details the development of connections in the client-server network. Two-way coupling between CFD solvers and Unity was demonstrated. Both OpenFOAM and COMSOL were tailored in the automated workflow. A case study was prototyped to assess the implementation including desktop, mobile, and virtual reality applications. The software packages developed in this study are freely available in the Supplementary materials of the digital publication.

3.1. Input from user application to CFD solver

Depending on the end-user device, the user can interact with the application through miscellaneous options. Touch screens, hand controllers, haptic devices and many others may be equipped to acquire inputs from the user. These interactions are directly adjusted in Unity regardless of the CFD solver.

3.1.1. Built-in parametric connections

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

User interaction with simulation data can be maintained using a case file that is exported from the CFD solver. A case file, also known as a system file, is a collection of sequential steps executed in the CFD solver written in a suitable API format. Any parameter in the case file can be externally adjusted. By importing the case file in the CFD solver, a simulation can be automatically run without any manual interferences. This approach enables a modular workflow to develop automated routines. Thus, the connection in the network was maintained based on case files exported from CFD solvers. First, a case file from a particular simulation was exported from the CFD solver. It was then embedded in Unity and linked to the user application's graphical user interface (GUI) for intended parameters. A file operation was ordered to process the user input in the relevant line and update the case file in runtime. In Unity, the operation was internally handled with a C# script, named as “ToTextFile.cs”. Finally, the updated case file was transferred to the server via “network-script” to perform a new simulation. Fig. 2 shows an example of the file operation in Unity to process input data into an OpenFOAM case file. A parameter-based approach was intrinsically taken into consideration to process the input from the user application.

Fig. 3 illustrates an overview of the implementation of both OpenFOAM and COMSOL case files in Unity. A modular connection was achieved without any hardcoded element in the dataflow. The “network-script” and the CDN settings in the game engine automate data transfer inside the network. It is noteworthy to mention that geometry models utilized in CFD simulations should be thoroughly logical, precise and stable to create and update a proper grid structure. Hand controllers and haptic devices can be still troublesome to accurately manipulate geometric features in the spatial domain (Huang et al., 2015; Shi et al., 2020). Therefore, the acquisition of input data through embedded user interfaces was mostly maintained with the parameter-based approach. This methodology sufficiently serves in the development of simplified, content-specific and flexible GUI for any CFD simulations, whereas non-expert users can readily adjust input parameters.

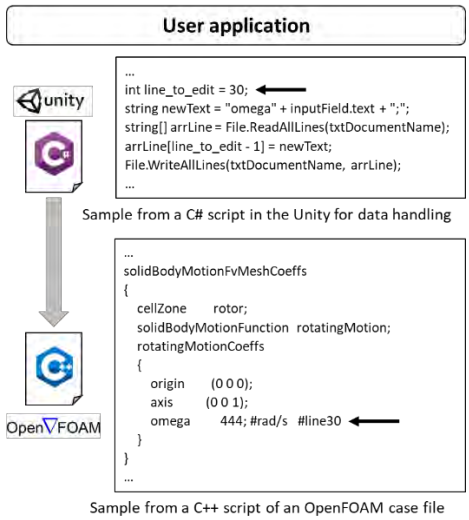


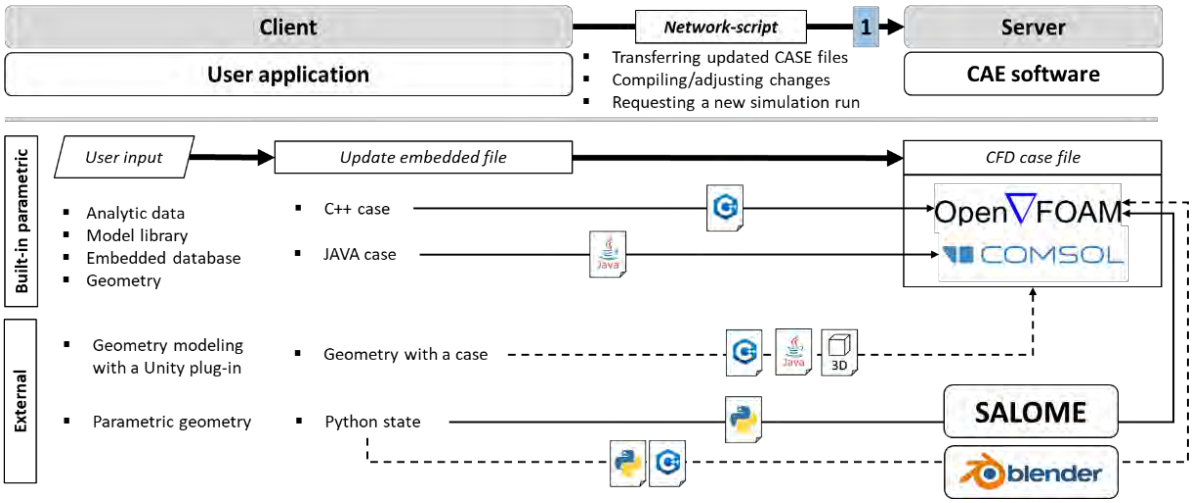
Fig. 2 Processing the input into OpenFOAM simulation case file which is embedded in the user application built in Unity.

3.1.2. Geometry and mesh specific external connections

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Skilled users may prefer an interactive freestyle fashion to manipulate geometric features. Although Unity does not provide any 3D modeling toolbox by default, the software has a modular structure supported by external toolboxes for 3D modeling. As can be seen in Fig. 3, we proposed two external connections for advance geometric operations in the system. The prior is for plug-ins such as ProBuilder and PIXYZ that can be imported into Unity for advanced runtime geometric operations. The manipulated geometry file should be exported in a 3D file format suitable to CFD workflows of COMSOL and OpenFOAM. The geometry is accompanied by a case file to integrate manipulations into the automated CFD routine.

COMSOL has a built-in geometry module to tackle geometric operations inside the software which can internally process both built-in parametric and external connections. In contrast, features in OpenFOAM, such as blockMeshblock and snappyHexMesh, enable a limited interaction to edit and update geometry directly in the case file, as handled with the parameter-based interaction. The geometry is generally hardcoded with predefined nodes and edges. This method can be problematic to process 3D and complex geometric modifications. Hence, the latter option, called the parametric geometry, is specifically dedicated to OpenFOAM within an open-source workflow to handle advanced geometric manipulations. An intermediate computer-aided engineering (CAE) software should be tied up to process geometric changes automatically. A complete connection was maintained by employing SALOME which is an open-source 3D modeling and meshing software. The software has an extensive Python API enabling various parametric operations to remotely reprocess geometry and meshing from a Python script. Initially, a geometry model with a grid structure was created from the ground up, and the project was saved as a *dump study* which collectively and sequentially writes all operations taken by developers into a Python script. As applied for built-in parametric connections, the Python script was embedded into the user application to process runtime interactions. This enables developers to prompt fully coupled geometry and meshing modules for OpenFOAM simulations. Any user-defined parameter processed in the Python script of a dump study can be automatically reproduced from the batch in the automated workflow. It is noteworthy that Salome exports meshes in UNV format that should be converted in OpenFOAM with *ideasUnvToFoam* utility. Besides, automation in the system can be straightforwardly expanded utilizing Python API of SALOME; for example, creating an automated grid convergence study.



Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Fig. 3 Built-in parametric and external operations to acquire the input from the user applications throughout modular and API-based connection. While continuous lines promote direct connections maintained in the study, dashed lines illustrates potential alternatives.

Furthermore, Blender, a 3D computer graphics software, can be considered to maintain a modular, API-based connection for geometric operations. This connection can convert geometry data to formats readable by Unity. Utilizing the Python API of Blender, an interoperable connection can be developed in which the input is processed in a Python state file embedded in the application. The connection requires external meshing software to recreate grid structures. In this study, instead of maintaining a working implementation, the connection with Blender is only to highlight the flexible structure of the two-way coupling strategy which enables developers to integrate and readily switch among various platforms.

3.2. Automated CFD routines and data extraction

Similar to calculations in the CFD simulation workflows, pre- and post-processing are also considerably time-consuming and iterative steps. Once an optimal workflow is decided by an analyst, pre- and post-processing steps turn into repetitive tasks that can be automatically executed. An automated CFD workflow with restricted options should be carefully prepared to control stability, precision and quality. The second connection in Fig. 1 points out to automated routines of OpenFOAM and COMSOL connected to the user application. Software-specific settings are essential to obtain inherently automated routines of each solver such as local and adaptive grid refinement, grid quality control, and many others commonly utilized in semi-automated CFD workflows.

Pre-processing, calculations, post-processing, and data extraction are the main steps sequentially executed in a typical CFD simulation. A fully automated CFD routine can be maintained with a script to execute each step from the relevant directory. Fig. 4 depicts automated routines for each solver with detailed workflow diagrams. The “server-script” manages the utilization of command scripts to automate the workflow; batch processing in Windows and bash processing in Linux operating systems (OS).

Automation of data extractions should be internally maintained in CFD solvers. As long as extracted CFD datasets are readable by ParaView, various data formats can be considered in the data processing software concerning data size, processing performance and quality. “Server-script” runs batch and bash scripts together with dedicated file handling operations to command executions in a row.

In order to automate the execution of modules in OpenFOAM, “Allrun” bash script was prepared to run each module in a user-defined fashion. Since the source code is not manipulated, no compilation is required in the system. OpenFOAM gives an output of each time-step separately in native format. A ParaView post-processing state was generated to automate the whole post-processing and data extractions tasks in the routine of OpenFOAM.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

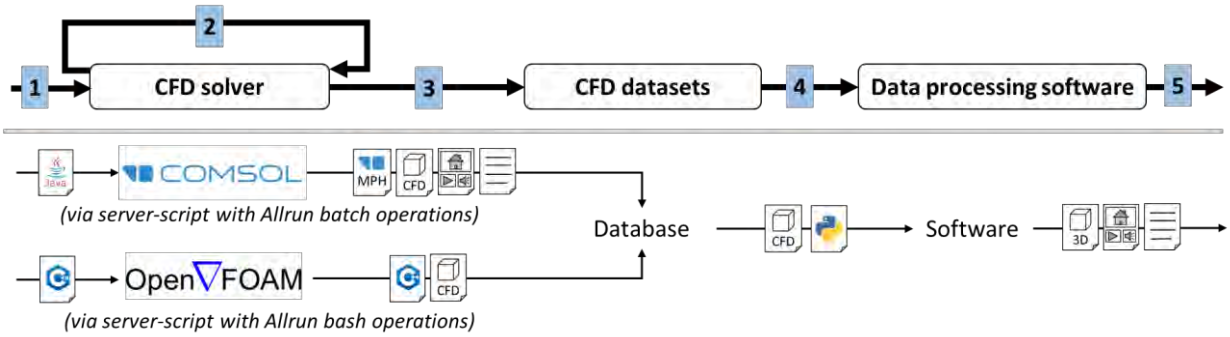


Fig. 4 Automation of CFD routines and processing of data throughout the modules in the server.

Unlike OpenFOAM, COMSOL only gives an MPH output file which cannot be directly processed in ParaView. Therefore, an extract-based data processing approach should start from the CFD solver. COMSOL has a post-processing module that can extract visual CFD data in VTU formats as a visualization toolkit (VTK) asset, thus enabling data processing in ParaView. All required orders should be set in the solver to extract and process necessary data through the automation routine. Automation was maintained with a JAVA file that is exported from COMSOL after completing a baseline simulation case. The JAVA file should be compiled with a suitable version of the JAVA software development toolkit (SDK) in the server. A CLASS file was automatically created after the compilation. The "server-script" then requests a new COMSOL project with the CLASS file and executes orders from batch without GUI. It should be noted that COMSOL is commercial software and comes with several prerequisites to protect data-in-use. All preferences in settings must be switched to authorizing external activities and interaction with changes while preparing the case files. The "server-script" should be accompanied by "Allrun" batch script on Windows to command COMSOL executions. Moreover, we utilized the *compact history* feature in COMSOL through which redundant steps in the development were discarded to create a JAVA file giving all steps straightforwardly.

3.3. Data processing software: integration of CFD data with game engine

The data generated with CFD post-processors is not compatible with the game engine; hence, it is a must to further process CFD data to data formats that are readable by cross-platform environments. A data processing methodology was previously published to integrate CFD data with game engines (Solmaz and Van Gerven, 2021). The study proposed an extract-based data processing strategy upon open-source, lightweight, modular and automated elements in the processing pipeline with software written in Python. It provides a remote data processing approach that makes any user device employable regardless of its computational power. A widened investigation was also performed among the most commonly used data formats to integrate CFD data with Unity. The methodology additionally presented a module to tune processing parameters based on data processing speed, size and quality. In the present study, we applied the same methodology to integrate, optimize and automate the data processing in the two-way coupling.

3.4. Data in game engine and user application

A remote connection between data and user application is the backbone of the automated two-way coupling. Once a user application is built in Unity with default settings, it behaves as standalone offline software. The content in the application is inherently static without any remote interaction. Unity can be

used to configure a CDN maintaining a remote connection between servers and clients. This way data can be streamed in the user application in runtime by avoiding default standalone built-in features. To develop a CDN with Unity, the user application should be linked to a server with a static connection during the cross-platform built-in. The following sections scrutinize the utility of features in Unity to generate a CDN in the client-server network.

3.4.1. Data normalization and representation

CFD visualizations, supplementary multimedia and analytic data are three data types processed and subsequently integrated into Unity. While the first two can directly be utilized in Unity after data processing software, the latter needs internal handling as shown in Fig. 5. Text-based files generally consist of raw data, which means that the information in a file should be split into parcels to be correctly interpreted. A simple data parsing algorithm with "getText.cs" script in Unity is applied to transform data in a readable type to retrieve and stream in user applications. Without data parsing, Unity displays all information simultaneously. On the whole, the parsing algorithm creates a grid of parcels, and numbers each parcel with a specific number to easily locate and stream data. Any changes in the text file are automatically updated, parceled and streamed in runtime.

Each simulation data requires a *GameObject* to be stored and represented in the user applications. A *GameObject* in Unity is a component that may contain varying information to create functional objects. For instance, a 3D simulation data is a *GameObject* comprising location in the spatial domain, 3D geometry, texture and datasets. Representation of simulation data, as in the form of *GameObjects*, should be manually settled only the first time. The information is automatically updated in the relevant *GameObject* if it is linked to the CDN.

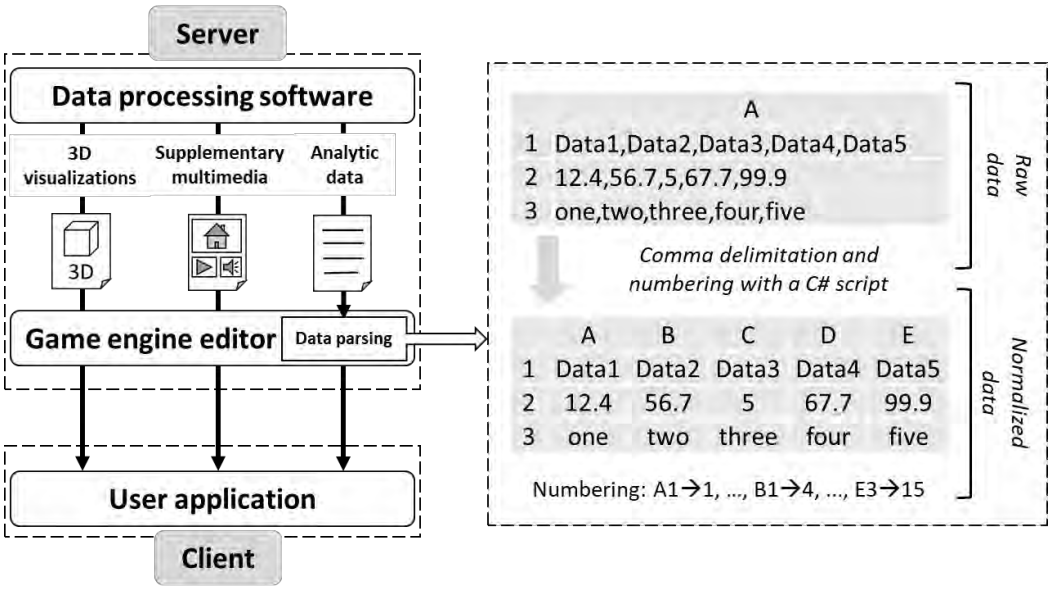


Fig. 5 An example of data normalization in Unity.

3.4.2. Cross-platform built-in and content delivery network

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Unity generates user applications with data stored in the project's asset directory during the built-in. No connection is maintained between data and user application by default. This results in standalone user applications that should be rebuilt with the game engine for any intended updates (Leskovsky et al., 2020). A dynamic connection between data and user application is one of the pillars of two-way coupling. Unity comprises well-oriented data management tools that are fundamental to host, manage and stream data remotely without any rebuild required. Fig. 6 illustrates an overview of the tools that were implemented in our work to develop a CDN in the client-server network.

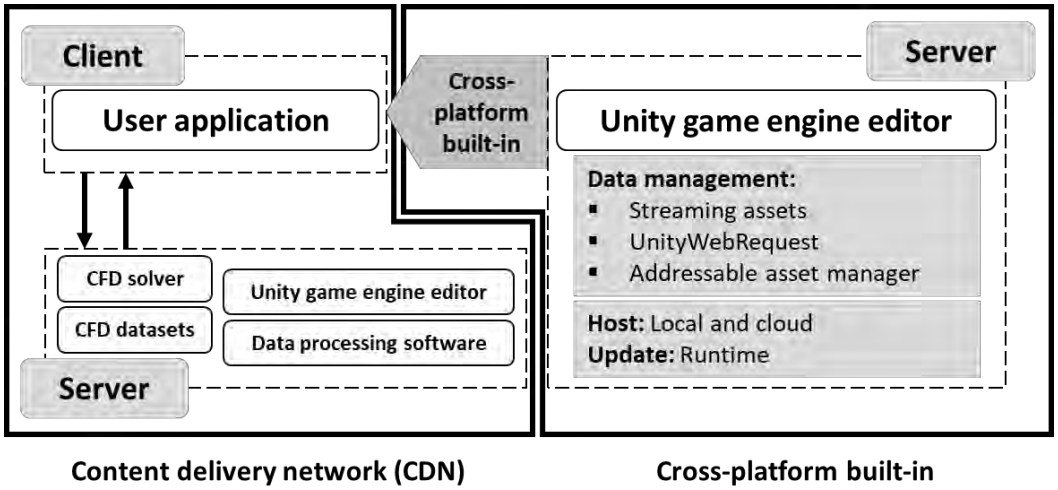


Fig. 6 Content delivery network and cross-platform built-in in the client-server network.

Data management in Unity can typically be classified as file-based, unity package, asset bundle manager, and addressable asset manager. The latter is the most advanced type and brings miscellaneous options to handle data in Unity. It manages memory in runtime, groups assets in catalogs, makes connections to load assets from remote serves, and bundles assets efficiently concerning data size, processing speed and quality. In this study, the connection between data and user application was mainly built with the addressable asset manager in Unity editor. Addressable options were activated for target GameObjects to build addressable content that can be renewed from a remote server. On occasion, streaming assets with UnityWebRequest were also considered to be nearly static, seldom, updated and simpler dataflow. Both approaches provide a modular system for composing HTTP requests and handling HTTP responses in runtime between client and server.

3.5. Dynamic post-processing in the client

Post-processing of CFD simulation data is a heavily interactive and iterative step taken manually by a competent analyst to present meaningful data. Once an optimal state is reached, the post-processing turns into a repetitive task that can be automated with a post-processing state file in the two-way coupling. Besides, additional interactive features in the user application may help non-expert users to increase comprehension of simulation data, as well as to provide flexible but restricted data exploration. In doing so we developed a dynamic post-processing data pipeline providing a connection between the client and data processing software in the server, as shown in Fig. 1.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Literature mainly demonstrated workflows to reprocess simulation data in the user applications: First, the native CFD data is processed in a CFD post-processor and results are exported in a dedicated format. Following, the exported file is transformed through the hardcoded methodology to integrate data with regard to the cross-platform. Finally, the file is exported in the cross-platform environment to reproduce data in the user application. This approach generally needs a lot of development work in case a change is intended to utilize another post-processing feature. Also, a suitable user device should be employed to handle comparable heavy data processing.

Our methodology differs from that in the literature. We do adhere to extract-based and modular features. Similar to the first connection in the client-server network, a case file-based dataflow was targeted, thus enabling a modular entity to maintain dataflow. The case file of the post-processing was embedded in the user application. Having updated the case file, the “network-script” sends the file to the server, in which the post-processing is performed, and streamed back to the client subsequently. A Python API-based implementation was achieved with open-source inter-connected modules of ParaView, VTK and Python. In essence, ParaView and VTK are widely used for advanced post-processing tools. Within the state file-based connection, any feature of these tools can be integrated into the user application through VTK classes, including case-specific post-processing methods. Basic post-processing entities, such as slicing, clipping, animating and color map labeling, were initially implemented in the user environment. We also built a simple animator to visualize transient simulation results in sequence. The user can play all time-steps, move between frames and create loops, as in a videocassette recorder.

3.6. Case study

A case study was performed to assess two-way connection with user applications. A client-server network was maintained using heterogeneous modules to easily access different resources. A Dell laptop with Intel(R) Core(TM) i7-8850U CPU @ 2.6GHz - 32 GB with Windows 10 and Ubuntu 14 OS was set up to run modules in the client-server network. Fig. 7 shows the implementation of the client-server network.

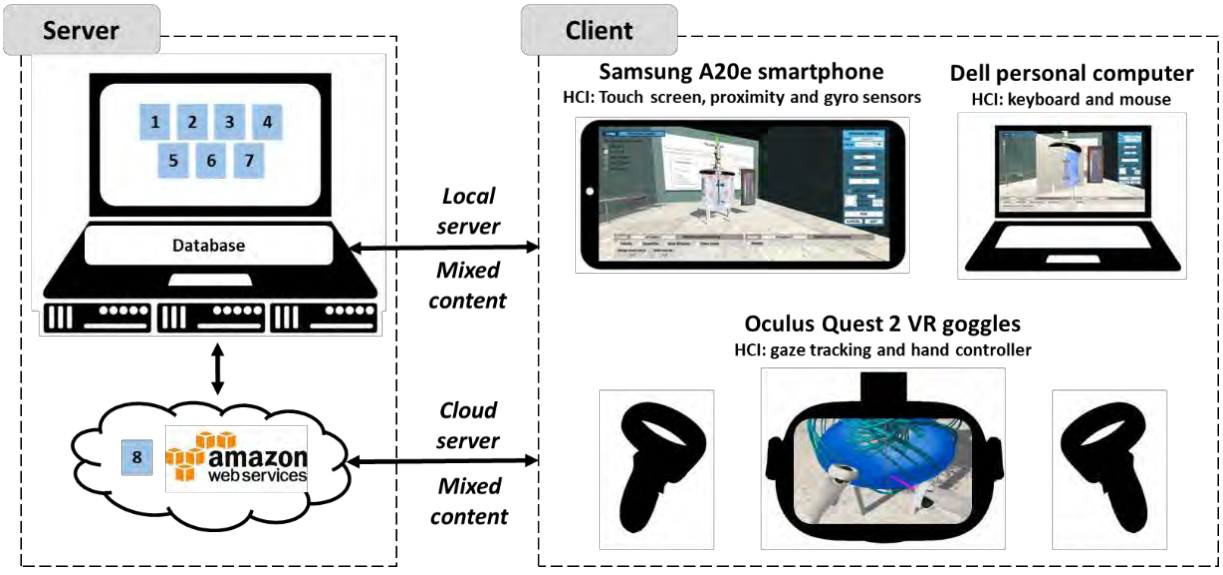


Fig. 7 Implementation of the client-server network and connected user applications with relevant HCI components.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

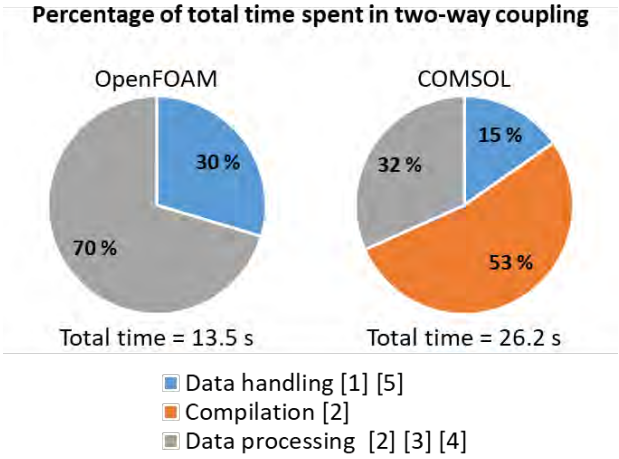
Both OpenFOAM v6 and COMSOL v5.6 were integrated into the user environment. CFD simulations were carried out to analyze a stirred tank reactor investigating a liquid mixing process. OpenFOAM simulation was performed to optimize baffle plate arrangement in the tank by customizing *mixerVessel2D* tutorial. A k-epsilon turbulence model was utilized to model fluid flow with 2D and steady-state arrangements. A computational grid was structured with 76800 quadrilateral cells. Likewise, simulations in COMSOL were calculated to compare different impeller characteristics utilizing *Mixer* application from COMSOL application gallery. A 3D transient simulation of the mixing tank was derived to assess mixing characteristics with radial and axial impellers. Fluid flow was modeled with the algebraic yPlus turbulence model and 103197 tetrahedron cells were processed in the computational grid.

User applications and CDN were developed with Unity game engine 2019.2.18f1. We maintained the connection using one local on the laptop and one cloud-based host from Amazon Web Service (AWS). Three executable user applications were developed; a mobile application for Samsung Galaxy A20e 32 GB with Android 10; a desktop application for the laptop with Windows 10; and a VR application for Oculus Quest 2 64 go. A video recording of the VR experience is available in the Supplementary materials.

4. Results & discussion

4.1. Overall system performance

The two-way coupling consists of several connected modules in the client-server network. Data is processed in the server and then streamed in the user application. It was revealed that the entire coupling can be managed with modular API-based connections. Fig. 8 shows the overall system performance of two-way coupling through the client-server network. The system performance was obtained as the average of 10 repetitions with a 1.4% maximum error rate of the processing time. Each automated CFD routine was ordered to process a package of simulation data with the same size consisting of a 3D model, image and analytic data. The total processing time between the client's request and the server's response takes less than half a minute, excluding calculations in CFD solvers. The results indicate the importance of handling all data processing work in the server instead of in the user devices. Only the lightweight, mobile-friendly data is circulated in the client-server network. In both cases, output data sizes were reduced from 60 MB to 3 MB thanks to the extract-based data processing approach, thereby requiring 20 times less memory in the end-user device.



Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Fig. 8 Overall system performance of the connection and comparison among CFD solvers; percentage of total time spent in two-way coupling.

The total processing time in OpenFOAM covers CFD data post-processing since ParaView can directly be applied to the native CFD datasets. In contrast, COMSOL performs post-processing inside the software. Visual extracts are then processed with the data processing software to integrate data into Unity. In addition, COMSOL requires a compilation before running a new simulation from the JAVA file. Therefore, automation of COMSOL tends to take longer than that of OpenFOAM, as shown in Fig. 8. If the time spent during compilation is counted as a part of the automated routine, both CFD solvers approximately take 13 seconds to maintain a two-way connection. Table 2 presents a chart with details of connections in the two-way coupling cases. Employing more powerful hardware in the server can further lower the processing time spent in each module. The system collects extract-based simulation results to develop a database in the meantime. If the request of the user is previously calculated and stored in a database in the server, in only a few seconds the user receives a response and streams data in the application. Scene from mobile, desktop and VR applications can be seen in Fig. 11 and Fig. 12, respectively.

Inevitably, interactive CFD simulations may take far longer time periods and a higher number of post-processing steps than what has been demonstrated in the implementation. It should be noted that educational simulations can be carried out for simplified but still accurate cases, especially targeting abstract examples. This may sufficiently serve CFD non-experts to learn the fundamentals of technical content and the concept of CFD simulations. As well, concerning generic but case-specific applications, reduced-order models (ROM) can be considered in the pipeline, which is becoming popular in the context of digital twins.

Table 2 The blueprint of automated two-way coupling with details on the system performance.

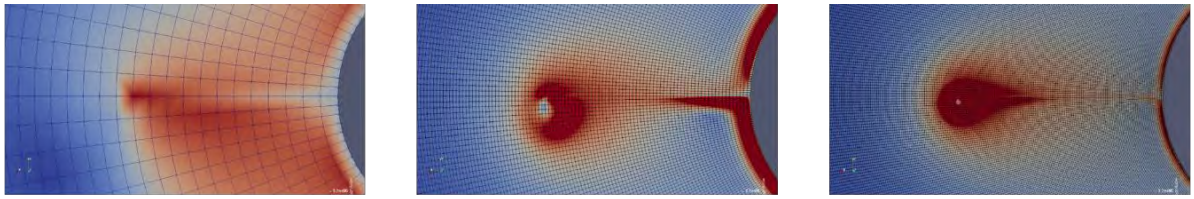
Connection	Parameters	User input [1]	Client to server [1]	Compile [2]	Automated CFD solver [2]	Data processing [3, 4]	Server to client [5]	Streaming in client [5]
OpenFOAM	Data type	OpenFOAM case	OpenFOAM case	-	OpenFOAM case	Mixed data	Mixed data	Mixed data
	Input data	text	C code	-	C code	C code	FBX; PNG; CSV	FBX; PNG; CSV
	Output data	C code	C code	-	C code	FBX; PNG; CSV	FBX; PNG; CSV	FBX; PNG; CSV
	Processing time (s)	<1	<1	-	268	9.5	<1	<1
	Output data size (MB)	0.004	0.004	-	60	3	3	3
	Handler	client	network-script	-	server-script	server-script	network-script	client
COMSOL	Data type	COMSOL case	COMSOL case	COMSOL case	COMSOL case	Mixed data	Mixed data	Mixed data
	Input data	text	JAVA code	JAVA code	JAVA CLASS	VTU	FBX; PNG; CSV	FBX; PNG; CSV
	Output data	JAVA code	JAVA code	JAVA CLASS	MPH; VTU; PNG; CSV	FBX	FBX; PNG; CSV	FBX; PNG; CSV
	Processing time (s)	<1	<1	13.9	2,487 + 6 (post-process)	2.3	<1	<1
	Output data size (MB)	0.121	0.121	0.085	60	3	3	3
	Handler	client	network-script	server-script	server-script	server-script	network-script	client

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

4.2. Effect of CFD model scale on data processing quality

In a traditional CFD data visualization pipeline, the visual data is recalculated each time from data points encoded in a native format. In contrast, in the present study, a data processing software is employed to integrate CFD visual data with Unity by transforming native CFD data to tessellated data formats. The processing creates textures and tessellated geometry models from native CFD data, resulting in drastically reduced data size and processing time. A user device can readily handle and stream processed data without any heavy calculation required.

A comparative study was carried out to probe further on the effect of the total number of grid elements in tessellated data size. Fig. 9 illustrates three different computational grid structures with varying cell numbers. It was observed that total numbers of elements in a grid do not show an influence on tessellated data size (Table 3).



Grid-1: 3072 cells Grid-2: 76800 cells Grid-3: 307200 cells

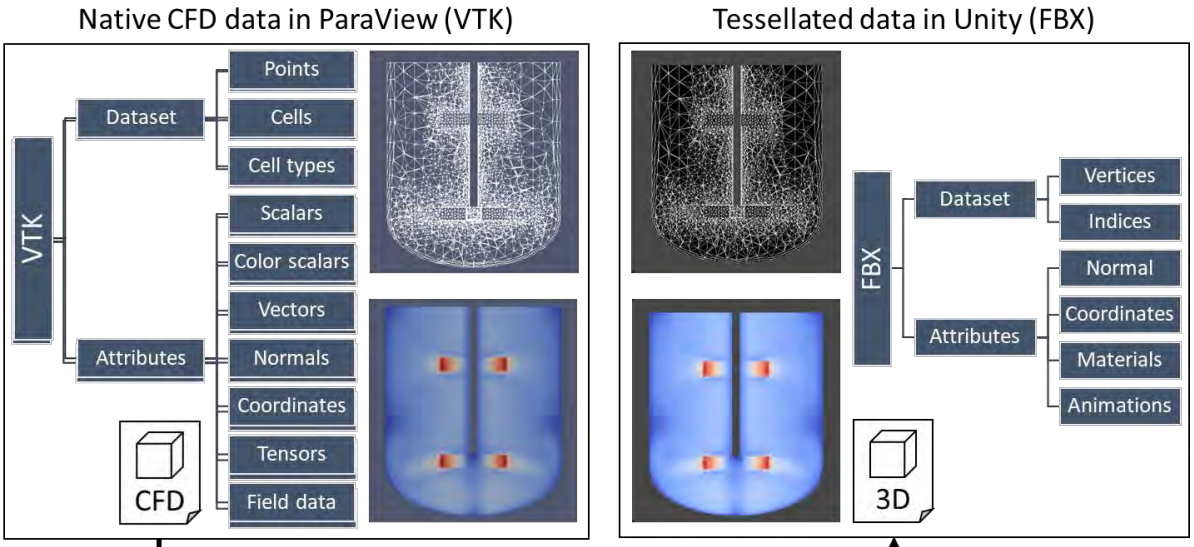
Fig. 9 Number of grid elements used in the comparison.

Table 3 The effect of CFD model scale on tessellated data size.

Parameter	Grid-1	Grid-2	Grid-3
Number of cells	3072	76800	307200
OpenFOAM file size (kB)	941	23100	95800
Data processing time (s)	2.55	10.05	33
FBX tessellated data size (kB)	3271	3271	3271

Another concern over the integration of CFD visual data with tessellated formats was reduced data quality on screen and resolution. To assess this, we compared the processing of ParaView in VTU format to tessellated data in FBX as shown in Fig. 10. The comparison highlighted that the grid structure utilized in the simulation is kept the same or tessellated during the data processing as long as triangulation is performed in the post-processor. A texture is created based on data points on the grid structure, thereby keeping the resolution equal. This concurs well with a recent study on the data quality of tessellated formats to integrate simulation data with Unity (Wang et al., 2020). Resolution of graphics should be adjusted in Unity with regard to the user hardware employed such as texture resolution from render scale and anti-aliasing filter.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570



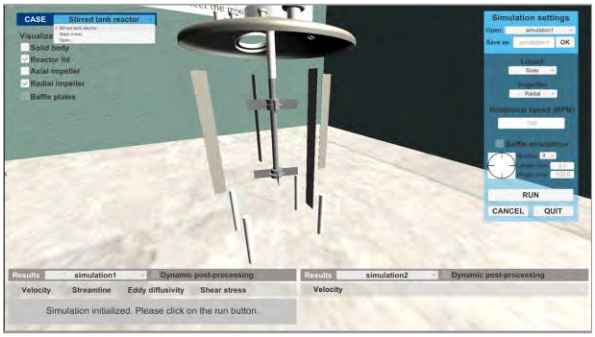
Data size	Native (MB)	VTK (MB)	X3D (MB)	FBX (MB)
COMSOL	60	0.6	4	0.3
Import in Unity	-	-	-	+

Fig. 10 Data resolution throughout processing and integration of visual CFD data into Unity.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

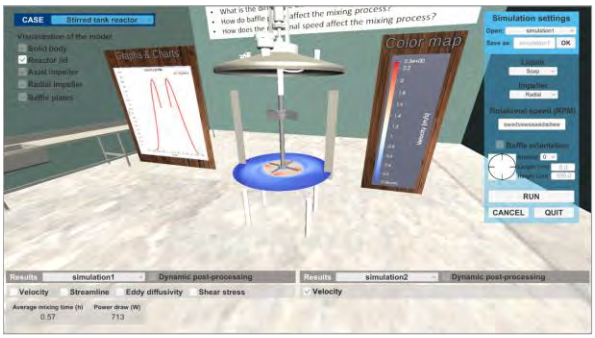
(a) Introducing learning task

(b) Visualization of geometry

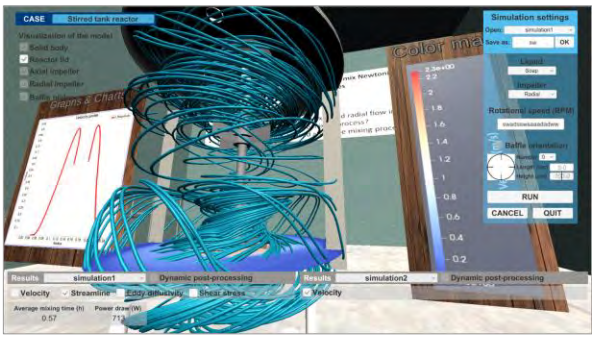


(c) Case load and GUI

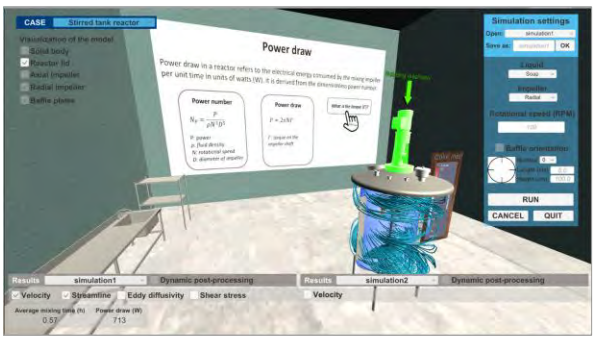
(d) Simulation with COMSOL



(e) Simulation with OpenFOAM



(f) COMSOL and OpenFOAM



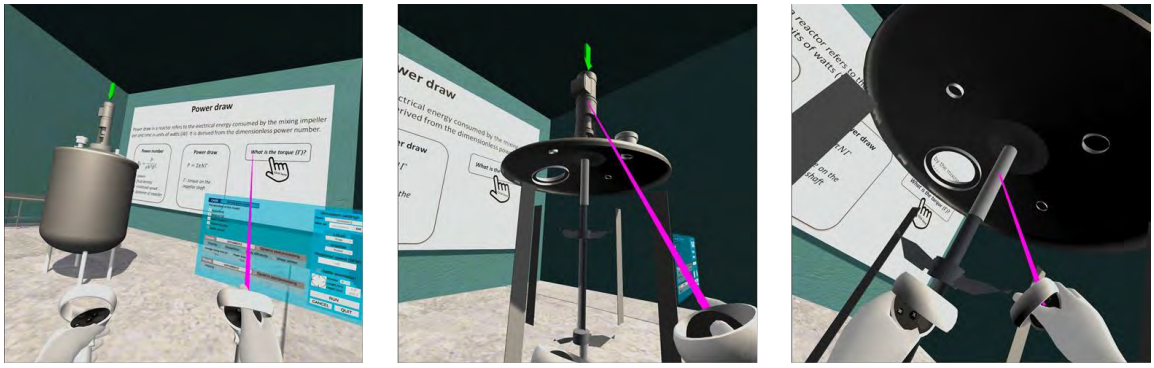
(g) Supportive and just-in-time information

(h) Dynamic post-processing with a clipping-plane

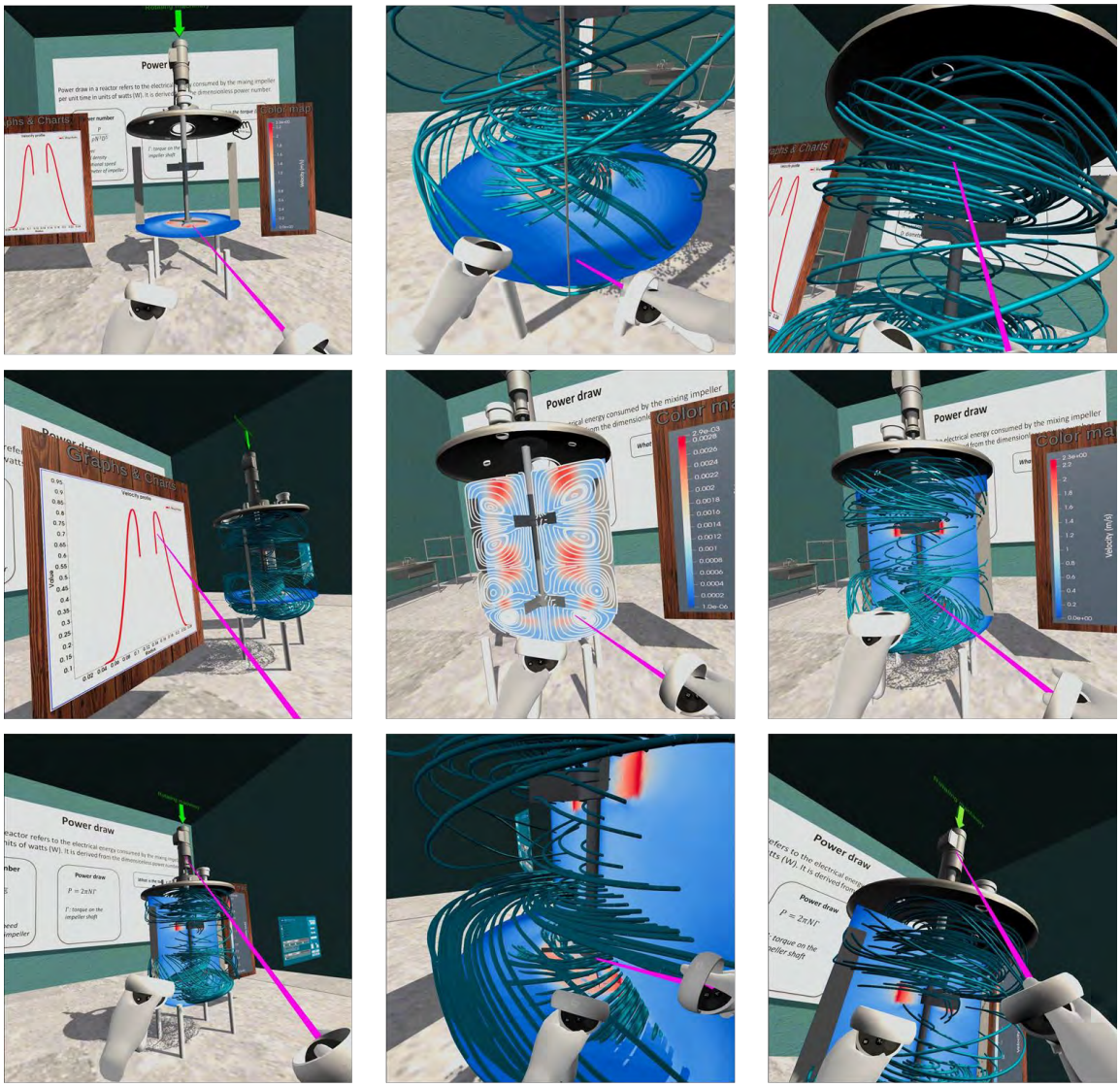
Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Fig. 11 Scenes from desktop and mobile user applications.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570



(a) GUI in VR and interaction with geometry



(b) Data exploration in VR

Fig. 12 Scenes from the user application in Oculus Quest 2 VR goggles with hand controllers.

4.3. Data recuperation in COMSOL

4.3.1. Data consistency and transient processing

Preliminary data extraction from COMSOL was previously mentioned in order to read data with ParaView. It was revealed that the extracts created in the design workflow should be kept in the working directory of COMSOL as a reference for the automated cases. Otherwise, even if the JAVA file processes a command to extract data during the processing, the software gives no input without references. Furthermore, it was observed that transient VTU extracts of COMSOL cannot be simultaneously imported in ParaView. The transient data should be piled with a PVD merger created with ParaView.

4.3.2. Processing streamline data

Streamline is one of the most used post-processing methods to visualize CFD data. It is formed from curves and splines derived from each cell point. A geometric extrusion is generally applied to extrude 3D structures in tube formats. COMSOL saves native CFD data in MPH format, which cannot be imported in ParaView without any preliminary processing in the software. To integrate the visual CFD data produced with COMSOL in Unity, COMSOL can export CFD visual extract in VTU format as an asset of VTK legacy, thus enabling data processing with ParaView. It was observed that VTU encodes only simulation data rather than geometric features; hence, the format does exclude all geometric features while extracting data from COMSOL, for instance, extrusion of streamline tubes from splines. An analysis was carried out to overcome this issue. The analysis revealed that Blender can extrude tube forms from bevel curves which would be an alternative way to recuperate streamline data in VTU format as shown in Fig. 13. Data processing software, presented in Section 3, comprises Blender as an intermediate data processor transforming CFD native data to tessellated formats. Therefore, data recuperation can be directly performed in the data processing module by adding the *if statement* conveyed with Listing 1.

```
scene = bpy.context.scene
scene.layers = [True] * 20 # Show all layers
for obj in scene.objects:
   if obj.type == 'CURVE':
      scene.objects.active = obj
      bpy.ops.object.mode_set(mode='EDIT')
      bpy.context.object.data.bevel_depth = 0.02
      bpy.context.object.data.bevel_resolution = 2
      bpy.context.object.data.fill_mode = 'FULL'
      bpy.ops.object.mode_set(mode='OBJECT')
```

Listing 1 Extrusion of steamline data in VTU data format with Blender using bevel modifier.

Moreover, streamlines in the 3D tube format do result in large data sizes and processing times. Extrusion of streamline with Blender can reduce data size and processing time by tuning the depth and resolution of 3D tubes. We carried out a study to elaborate on our tuning approach as shown in Table 4. Different data formats and processing pipelines were compared based on extrusions in ParaView and Blender. FBX opted for the final tessellated format. Interestingly, extrusion in ParaView drastically increased metadata produced along the process. Even though it did not affect the size of the final data format FBX, the process results in large metadata temporarily being kept in the server. Extrusion of streamline in Blender (#1, #2,

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

#3) requires lesser meta-data size and processing time than other workflows. This result confirms the usefulness of Blender in data recuperation. The same processing methodology can be applied to tune point clouds and vector-based processing. In addition, data processing workflows of GTLF, OBJ and PLY formats without extrusion in ParaView failed to export simulation data from ParaView to Blender.

Streamline data in curve without geometric extrusion

Streamline data in curve with bevel modifier in Blender

Fig. 13 Streamline data recuperation from VTU file; (left) without geometric extrusion; (right) bevel modifier in Blender.

Table 4 Streamline data processing pipeline and comparison among workflows.

Workflow	ParaView				Blender			
	Size (MB)	Extrusion	Time (s)	Export format	Extrusion	Time (s)	Size (MB)	Export format
#1	15.3	no	0.5	X3D	yes (r=12 mm) (resolution =2)	5.93	21.79	FBX
#2	15.3	no	0.5	X3D	yes (r=5 mm) (resolution =2)	6	22.07	FBX
#3	15.3	no	0.5	X3D	yes (r=5 mm) (resolution =1)	4.6	16.6	FBX
#4	17.8	yes (r=12 mm)	0.3	GLTF	no	6	43	FBX
#5	90	yes (r=12 mm)	2.7	OBJ	no	104.8	21	FBX
#6	152	yes (r=5 mm)	4.86	X3D	no	19.7	21.74	FBX
#7	152	yes (r=12 mm)	4.93	X3D	no	19.6	21.92	FBX

4.4. Dynamic post-processing

We developed a connection for dynamic post-processing in the user device as shown in Fig. 11h. Giving a flexible, dynamic but thoroughly constrained post-processing environment can assist non-expert users to grasp knowledge and deeper data exploration. Dynamic post-processing performances of OpenFOAM and COMSOL by means of methodology and data processing time are shown in Table 5. If the relevant post-processing is available in the database, marked with #1 in Table 5, the system only takes a couple of seconds to stream data in the user application.

Table 5 System performance of dynamic post-processing in the client.

Connection	Dynamic processing approach	Client-server network	Total processing time (s)
OpenFOAM to Unity	#1 Already processed and stored in the database	6-5	5.5
	#2 Processing required	6-5	15

COMSOL to Unity	#1 Already processed and stored in the database	6-5	3.9
	#2 Processing required (from VTU)	6-5	6.3
	#3 Processing required (from MPH)	1-2-3-4-5	48

The dynamic post-processing in OpenFOAM is carried out with either native CFD data or data extracts in VTM format. User input from the client is processed to the state file and then sent to the server. Developers can create interactive elements for any intended parameters with ParaView API through the state file. This technique shows a fast and robust way to post-process data with user interference. Since Native OpenFOAM data is directly processed with ParaView, any specific need can be embedded into the game engine with a state file. This provides a wide range of utilities to process data thoroughly including slicing, clipping, animating, as well as any case-specific post-processing with VTK components such as vtkFilters. The same approach can be utilized for available VTU extracts from COMSOL, which is highlighted with #2 in Table 5. Data is rapidly processed and subsequently streamed in the user application without including COMSOL and MPH files in the pipeline.

The technique requires a complementary workflow for COMSOL in case the intended dynamics post-processing does not have a VTU extract available in the database. In this approach, user input should be directly processed by COMSOL to extract relevant parts of the data. As mentioned previously, automation of COMSOL was maintained with the JAVA file in the client-server network. The software executes each line in the file subsequently from the ground up including pre-processing, calculations, post-processing and data extraction. No matter what is processed in the JAVA file, the procedure repeats the entire cycle from the first line. To prevail over this redundant process in dynamic post-processing, we proposed an alternative technique. In contrast to the JAVA API file-based connection, the system directly executes the user input with the “server-script” using the COMSOL MPH file from the batch. The workflow, therefore, can bypass steps until the post-processing. The “server-script” finds relevant native COMSOL data in MPH format from the database and processes the intended parameters with “edit” through the command given in Listing 2.

```
"C:\...\comsol.exe" -edit C:\...\mixing3d.mph -Dcs.canvascolor=150,0,0,0,150,0
```

Listing 2 An example of the command for running dynamic post-processing with COMSOL from batch.

Alternatively, COMSOL has a *method* feature to keep on recording actions taken by analysts in the software. Creating a custom method in COMSOL automates user-defined settings that can be executed later on calculation from batch without GUI interventions. The “server-script” can perform the execution from the following command line added in the batch file as in Listing 3.

```
"C:\...\comsolbatch.exe" -inputfile C:\...\mixing3d.mph -methodcall methodcall1
```

Listing 3 An example of the command for running dynamic post-processing with a method created in COMSOL from batch.

The main pitfall of dynamic post-processing with COMSOL from an MPH file is the processing time which takes more than half a minute, comparably higher than other processing discussed above. Another way for time-consuming dynamics post-processing techniques is to extract slices in sequence and keep them in the server with VTU format; therefore, the network does not need to run COMSOL. Since slices are 2D or iso-surfaces, not too much memory is allocated in the database.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

4.5. Asset management in Unity to create a content delivery network

A CDN was generated to remotely update simulation results in the user applications. A built-in is only one-time required in Unity to develop a user application with CDN settings. Both streaming assets with *UnityWebRequest* and addressable asset manager were utilized to make connections for CDN within Unity. Either free or paid cloud store facilities can be implemented in CDN as long as a static URL is provided for streaming assets and addressable asset manager. In the present work, we allocated a cloud server from Amazon Web Service (AWS) to host and distribute data in the client-server network. A local host was also created to experience how a personal computer can be transformed into a webserver instead of using the AWS system. Both servers were implemented into the CDN developed with Unity using streaming and addressable asset system as shown in Fig. 14.

Streaming assets with *UnityWebRequest* can create a remote connection for any game object in the game engine editor. It was found as useful due to the simpler workflow, a very short C# script generally provided by Unity. The connection should separately be maintained for each object in the digital environment. The method is incapable of managing multiple data in the client simultaneously.

Compared to the streaming assets, the addressable asset manager brings several advantages to build sustainable, long-lasting CDNs comprising several high-level operations to manage data and dataflow. The implementation revealed that addressable asset manager increases asset building and loading speeds. It also helps with runtime memory management in the user application. Category creation and labeling are important features that come with the management system to handle multiple data types. The system piles assets in bundles of 3D visuals, supplementary multimedia and analytic data for each simulation case. Not only *GameObjects* but also scenes in Unity can be remotely deployed within the addressable system. This feature facilitates developers to easily switch between entirely different simulation cases in the user application as shown in Fig. 11c.

Moreover, the addressable asset manager gives uncompressed, LZ4 and LZMA options to store assets with bundles comprising mixed data formats. A further assessment was carried out to understand the data encoding structure of the management system. An asset bundle from a particular simulation consisting of a 3D model, multimedia image and analytic data was utilized. The result showed that LZ4 and LZMA data compression methods can reduce asset bundle size 1.58 and 3.05 times, respectively, compared to the uncompressed format. In terms of data processing time, no significant difference was identified between the uncompressed and LZ4 methods to process and stream data in the user application. However, LZMA resulted in some time delay during the decompression of the asset bundle.

These results offer compelling evidence for the addressable system; however, due care must be exercised to implement the addressable data manager in a two-way connection. The game engine should be employed as an intermedia data processor to build and manage asset bundles before streaming in the user application. This brings 5 s of data processing time in the game engine, labeled with connection 8 in the client-server network (Fig. 1). The bigger the total file size gets, the longer the data processing takes. Despite the additional processing work and developer expertise required to apply addressable asset manager, it considerably aids in developing long-lasting digital environments with advanced management features and robust connections, especially if mixed content is an essential trait.

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

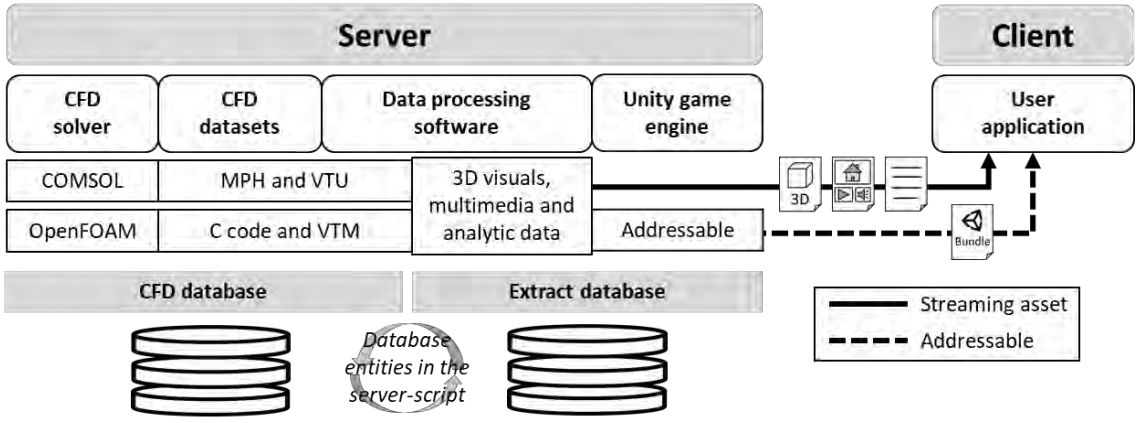
The CDN adheres to a network to stream data in runtime from a remote server. Although it can automate dataflow and manage data in applications, it also requires the continued support of servers and complicated software development workflows. Overall, our assessment is that a blended use of streaming assets with *UnityWebRequest* and addressable asset manager serves best to promote optimal ways to develop CDN with Unity. Beyond education, training, and broad communication; engineering design and analysis would benefit from the cross-platform integrity of CFD simulations through the CDN in the client-server network. This might open gates for interoperable co-simulation environments as shown in Fig. 11f, as well as the incorporation of digital twins with interactive CFD simulations for fast prototyping. Our work demonstrated a promising practice on the automation and management of CFD simulations in multiplatform environments (Fig. 11 and Fig. 12).

4.6. Data and database management in the client-server network

Both CFD solvers and Unity are structured with dedicated data management systems; hence, there is no need to control the internal circulation of data in these software. However, preliminary findings showed that the data in the two-way coupling should be handled with an external system with sub-entities due to strictly different data types in use. We proposed two distinctive containers under the database that are set in the server to manage extracts and CFD data as shown in Fig. 14.

The extract database is driven with server modules to operate the data streamed in the client. Here multiple data formats, so-called mixed content, are dealt with as briefly explained in the previous sections. In general, extracts are kept as they are without any management, except documenting a particular simulation data in the same directory. Notably, a data management system dedicated to mixed content can provide a robust, long-lasting network to circulate simulation data between client and server. It was revealed that Unity provides an “addressable data management tool” to store, compress, retrieve and categorize the mixed content utilized in the user application through CDN. An analysis of the utility of the tool was elaborated in the previous sections.

The CFD database is utilized for storing and retrieving CFD data from multiple solvers. The results indicated that COMSOL encodes and saves entire simulation data in MPH format; hence, keeping only MPH files in the database is adequate for managing simulation software in a two-way connection. Additionally, VTK outputs can be hosted in the database if a dynamic post-processing feature is entailed in the workflow.



Storage and retrieval of data in the client-server network

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

Fig. 14 Automated two-way coupling with distinctive databases; streaming assets and addressable workflows to maintain remote connection in the client-server network.

On the other hand, OpenFOAM keeps the case file apart from the data and outputs each time step in a separate directory. A study previously investigated the interoperability of different data models to store and retrieve OpenFOAM simulation data. The study suggested that the VTM data model, among several other models supported by ParaView, could be an optimal choice to store and retrieve OpenFOAM data for multiplatform utilities (Solmaz and Van Gerven, 2021). Our further examinations in this study revealed that data compressing models LZMA, LZib and LZ4 in ParaView enable a considerable reduction in data size while keeping the processing time almost identical to VTM without compressing method. VTM with LZ4 performed the optimal analytics with a 68% reduction in data size and 0.3 s of processing time. A detailed comparison can be seen in Fig. 15. This method reduces the data size by orders of magnitude in a considerable period of time. In this study, we utilized VTM as a terminal data format to store and retrieve CFD datasets in the two-way coupling.

All database relevant jobs are executed with dedicated sub-entities in the “server-script”. Metadata produced along the coupling are temporarily kept until the target data is obtained. Data in the client is handled by the user application. The results of this study cannot be taken as evidence for all automated CFD simulation workflows. It is worth mentioning that several data formats can interchangeably be considered based on import and export features of the post-processing software. A preliminary assessment is a must to understand system database characteristics for multiplatform utilities. Although exploratory, this study may offer some insight into database structures. Two distinctive sub-entities might be necessary as long as extract-based data processing is applied in the integration of CFD data with Unity.

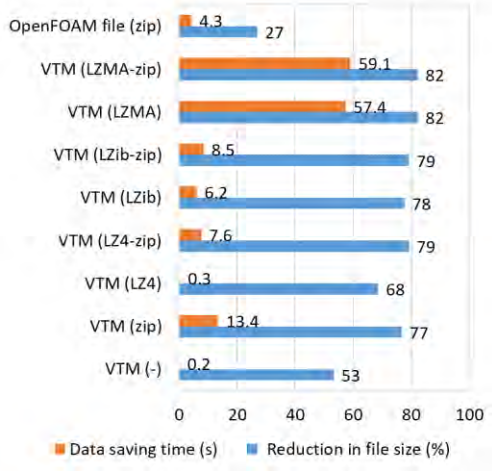


Fig. 15 Assessment on the VTM model to store and retrieve OpenFOAM simulation data.

4.7. Some aspects of practical implementation

4.7.1. Necessary financial and human resources

Content creation, network and hardware are the fundamental pillars of two-way coupled systems. To develop such VR applications presented in the study, financial needs and competent human resources are required.

In order to minimize the financial needs, our work proposed multiple solutions to facilitate development workflows with open-source software. The workflow with OpenFOAM comprises only open-source utilities that can be freely integrated into the system without any copyright issues. In contrast, COMSOL is a commercial CFD solver, thereby requiring a license to produce CFD data. Nevertheless, the software comes with various supplementary resources to guide users to effortlessly perform CFD simulations within an advanced GUI. Neither user guides nor a built-in GUI is available for OpenFOAM barring community contributions. Developers should choose the convenient CFD software upon their expertise in CFD and programming. The rest of the software in two-way coupling are structured towards open-source utilities comprising well-documented resources provided within this study, as well as publicly available user guides. In addition, a powerful workstation is generally a necessity to perform CFD simulations and host the server in two-way coupling at the back end. Either university resources or in-house hardware can be allocated to set the system up. Also, cloud-based solutions may be a useful aid to deal with remote and computationally intensive systems. For offline and standalone applications, in which the CFD data is stored in the user hardware, there is no need for a server, thereby reducing the cost of hardware facilities. Furthermore, by means of end-user hardware, Oculus VR goggles cost around € 400 at the time this study was conducted in 2021. The VR market is dynamic and composed of several brands for which Unity can also build the same applications with cross-platform utilities. Developers may easily switch to other VR hardware considering financial and technical aspects. Alternatively, a smartphone version of the VR application with Google Cardboard might be a low-priced counterpart, as well as an accessible technology. It is plausible that smartphone-based solutions adversely limit the effect of the immersion as they lack peripheral devices and relevant HCI components.

A diversified human resource is imperative to set up VR applications with CFD simulations due to the interdisciplinary structure of the study. Collaborative work should be prompted between developers and practitioners to overcome design and operational challenges. Our work includes a development strategy with provided tools that can be replicated by volunteers. We tried to divert conventional workflows into modular, easy-to-implement and coding-free utilities. Notably, proven expertise in CFD and computer science can drastically diminish the workload in the development. In addition, the design of educational digital environments should be critically examined with university lecturers to obtain the optimal user experience for targeted student groups. To sum up, it seems likely that composing a project group of people with diversified competencies may smooth the way for the development. Initial user prototypes can be roughly made ready in a couple of weeks. Agile development cycles with preliminary user studies are also strongly recommended to tune the user experience in VR environments.

4.7.2. Insights on educational practices

CFD simulations can bring heuristic solutions for fluid flow and multiphysics problems that are difficult and costly to study with physical experiments. Chemical engineering comprises several abstract concepts that can be intuitively examined with CFD simulations. Integration of these simulations in user-friendly VR environments may help students perform virtual experiments while learning from meaningful simulation content. VR applications can also leverage cognitive skills with advanced HCI, whereas students' motivation can be augmented within user-friendly, fun and engaging environments. This may allow both students and lecturers to benefit from CFD simulations while making them accessible in the early stage of higher education. VR applications have a wide range of scalability blending advanced learning content, technology and pedagogy in a unique way. This can be helpful for lecturers to divert the user experience

upon the needs of their target group from edutainment over stimulating academic interest to advanced visualizations. In addition, real-time feedback and data collection are other positive aspects of VR applications. Lecturers may exploit built-in assessments by keeping track of user actions in the VR application in the context of learning analytics.

Notably, VR applications can be blended with traditional lab courses being either complimentary or supplementary to physical experiments throughout content-wise rich educational practices. This may boost students' interaction with technical content whereas lecturers may touch upon challenging concepts that are not possible with current educational technologies. Interestingly, CFD simulations in the VR environment can be readily transformed into augmented reality (AR) applications (Solmaz et al., 2021). This may enable embodied collaborative tasks that students can experience and take while physically being present in the laboratory. For instance, transport phenomena in microfluidics with an AR experience, as a complementary tool for a lab course, may leverage both cognitive and behavioral aspects of educational practices. For verified and validated simulation workflows, neither lecturers nor students have to know the operational aspects of CFD simulations at the back end. Our discussions are restricted with supplementary and complementary utilization of such applications in lab courses through an act of balance that should be taken into account by lecturers. Replicating the entire lab experience with a virtual environment should be treated with utmost caution from a pedagogical point of view.

Inevitably, utilization of VR applications in education can be restricted - even avoided - because of the struggles both students and lecturers may face. From the lecturer's point of view, accessing the required amount of VR goggles for an entire classroom can be found financially challenging. In addition to this, the development, implementation, post-production and maintenance of such digital applications may increase the workload on lecturers. Therefore, a good way to avoid these hurdles is to compose a competent project group. Aside from technical concerns over the implementation of VR in classrooms, the impact of VR on students' health, so-called VR sicknesses, should be seriously taken into consideration by developers and practitioners. Both physical and cognitive harms may potentially arise from VR applications due to hardware- and software-induced deteriorations (Kourtesis et al., 2019). Students, who are vulnerable to VR environments, can develop negative side effects such as dizziness, disorientation, seizures, nausea and eye soreness. A meta-analysis on VR sicknesses revealed that the intensity of negative side effects is undeniably influenced by digital content, visual stimulation, locomotion, exposure times and user characteristics. Notably, the study illustrated the importance of simplified digital content and relatively low exposure times to tackle the negative implications of VR environments (Saredakis et al., 2020). Lecturers should keep an eye on students to comprehend their experience and identify major inadequacies preventing them to adopt new technologies. Performing technology acceptance studies within the preliminary prototypes may hint at these drawbacks to be avoided in the early stages of implementations. Recent publications delegate both developers and practitioners to alleviate negative side effects and ensure health and safety standards in VR experiences (Kourtesis et al., 2019; Saredakis et al., 2020). Furthermore, a desktop version of the VR application, such as a computer game, can be built in Unity in order to avoid grave aspects of virtual reality for vulnerable individuals. Though immersive features of VR are diminished in the desktop version, students and lecturers can still benefit from the user-friendly educational digital application with CFD simulations.

5. Conclusion and future work

Post-print version of Solmaz, Serkan, and Tom Van Gerven. "Interactive CFD simulations with virtual reality to support learning in mixing." Computers & Chemical Engineering (2021): 107570.
DOI: https://doi.org/10.1016/j.compchemeng.2021.107570

This study presents a versatile development methodology to implement interactive CFD simulations with cross-platform environments such as desktop and virtual reality applications. We developed a client-server network to maintain a two-way connection between CFD solvers and Unity, with promising applicability to employ different client user devices without any computational demand. Essentially, the connection is a great enabler to develop technically advanced learning environments with CFD simulations, as well as digital authoring tools for educational technology developers. In addition, the implementation of multiple CFD solvers simultaneously in the game engine formed a co-simulation environment. Not only education but also engineering can benefit from the cross-platform integrity of CFD simulations, for example, communication of CFD simulations to broad engineering communities. Beyond, the system might be beneficial to settling up digital twins, if on-site connected interactive CFD simulations are added value for operational purposes.

Future studies will tackle technical content to be progressively incorporated from macro- to micro-mixing, thus providing a widened tool to teach transport phenomena. More functional environments will be deployed facilitating well-designed GUI and HCI components. Qualitative and quantitative user assessments will be performed to determine the effectiveness of the VR applications. We also intend to extend the user environment with inquiry- and game-based learning approaches to profit more from educational theories to motivate learners and trigger their curiosity.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project has received funding from the European Union's EU Framework Programme for Research and Innovation Horizon 2020 under Grant Agreement 812716. This publication reflects only the author's view exempting the community from any liability. Project website: https://charming-etn.eu/. The authors would like to thank Dr. Liesbeth Kester and Dr. Bert Slof for their insightful comments to help design the learning environment.

Appendix.

Supplementary data associated with this article can be found, in the online version, at [DOI].

A.1. Video recording from the VR experience

A video recording of the user experience in the VR environment is available in the online version. It can be also found on https://www.youtube.com/watch?v=5Khu_bZhtZI.

A.2. Source code

Codes and packages utilized in this work are freely available on GitHub:
https://github.com/sersolmaz/CFD-CDN-UNITY.

References

Dahash, A., Ochs, F., Tosatto, A., 2019. Co-Simulation of Dynamic Energy System Simulation and COMSOL Multiphysics®, in: Proceedings of the 2019 COMSOL Conference. Cambridge, United Kingdom.

Duque, E.P., Imlay, S.T., Ahern, S., Guoning, C., Kao, D.L., 2016. NASA CFD Vision 2030 Visualization and Knowledge Extraction: Panel Summary from AIAA AVIATION 2015 Conference, in: 54th AIAA Aerospace Sciences Meeting. Presented at the 54th AIAA Aerospace Sciences Meeting, American Institute of Aeronautics and Astronautics, San Diego, California, USA. https://doi.org/10.2514/6.2016-1927

European Commission (Ed.), 2020. Digital education action plan 2021-2027. Publications Office of the European Union.

European Commission (Ed.), 2017. Digital Transformation Monitor: Augmented and Virtual Reality, Internal Market, Industry, Entrepreneurship and SMEs.

European Commission (Ed.), 2011. Supporting growth and jobs: an agenda for the modernisation of Europe's higher education systems. Publications Office of the European Union, Luxembourg.

Fernandez Rivas, D., Boffito, D.C., Faria-Albanese, J., Glassey, J., Afraz, N., Akse, H., Boodhoo, Kamelia.V.K., Bos, R., Cantin, J., (Emily) Chiang, Y.W., Commenge, J.-M., Dubois, J.-L., Galli, F., de Mussy, J.P.G., Harmsen, J., Kalra, S., Keil, F.J., Morales-Menendez, R., Navarro-Brull, F.J., Noël, T., Ogden, K., Patience, G.S., Reay, D., Santos, R.M., Smith-Schoettker, A., Stankiewicz, A.I., van den Berg, H., van Gerven, T., van Gestel, J., van der Stelt, M., van de Ven, M., Weber, R.S., 2020a. Process intensification education contributes to sustainable development goals. Part 1. Education for Chemical Engineers 32, 1–14. https://doi.org/10.1016/j.ece.2020.04.003

Fernandez Rivas, D., Boffito, D.C., Faria-Albanese, J., Glassey, J., Cantin, J., Afraz, N., Akse, H., Boodhoo, K.V.K., Bos, R., Chiang, Y.W., Commenge, J.-M., Dubois, J.-L., Galli, F., Harmsen, J., Kalra, S., Keil, F., Morales-Menendez, R., Navarro-Brull, F.J., Noël, T., Ogden, K., Patience, G.S., Reay, D., Santos, R.M., Smith-Schoettker, A., Stankiewicz, A.I., van den Berg, H., van Gerven, T., van Gestel, J., Weber, R.S., 2020b. Process intensification education contributes to sustainable development goals. Part 2. Education for Chemical Engineers 32, 15–24. https://doi.org/10.1016/j.ece.2020.05.001

Fukuda, T., Yokoi, K., Yabuki, N., Motamedi, A., 2019. An indoor thermal environment design system for renovation using augmented reality. Journal of Computational Design and Engineering 6, 179–188. https://doi.org/10.1016/j.jcde.2018.05.007

Ge, W., Guo, L., Liu, X., Meng, F., Xu, J., Huang, W.L., Li, J., 2019. Mesoscience-based virtual process engineering. Computers & Chemical Engineering 126, 68–82. https://doi.org/10.1016/j.compchemeng.2019.03.042

Ham, Y., Golparvar-Fard, M., 2013. EPAR: Energy Performance Augmented Reality models for identification of building energy performance deviations between actual measurements and simulation results. Energy and Buildings 63, 15–28. https://doi.org/10.1016/j.enbuild.2013.02.054

Harwood, A.R.G., 2019. GPU-powered, interactive flow simulation on a peer-to-peer group of mobile devices. Advances in Engineering Software 133, 39–51. https://doi.org/10.1016/j.advengsoft.2019.04.003

He, Z., You, L., Liu, R.W., Yang, F., Ma, J., Xiong, N., 2019. A Cloud-Based Real Time Polluted Gas Spread Simulation Approach on Virtual Reality Networking. IEEE Access 7, 22532–22540. https://doi.org/10.1109/ACCESS.2019.2893919

Huang, J.M., Ong, S.K., Nee, A.Y.C., 2017. Visualization and interaction of finite element analysis in augmented reality. Computer-Aided Design 84, 1–14. https://doi.org/10.1016/j.cad.2016.10.004

Huang, J.M., Ong, S.K., Nee, A.Y.C., 2015. Real-time finite element structural analysis in augmented reality. Advances in Engineering Software 87, 43–56. https://doi.org/10.1016/j.advengsoft.2015.04.014

Kim, M., Yi, S., Jung, D., Park, S., Seo, D., 2018. Augmented-Reality Visualization of Aerodynamics Simulation in Sustainable Cloud Computing. Sustainability 10, 1362. https://doi.org/10.3390/su10051362

Kim, R., Hong, S., Norton, T., Amon, T., Youssef, A., Berckmans, D., Lee, I., 2020. Computational fluid dynamics for non-experts: Development of a user-friendly CFD simulator (HNVR-SYS) for natural ventilation design applications. Biosystems Engineering 193, 232–246. https://doi.org/10.1016/j.biosystemseng.2020.03.005

Kirschner, P.A., Sweller, J., Clark, R.E., 2006. Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. Educational Psychologist 41, 75–86. https://doi.org/10.1207/s15326985ep4102_1

Konrad E. R., Boettcher., Behr, A.S., 2020. Teaching Fluid Mechanics in a Virtual-Reality Based Environment, in: 2020 IEEE Global Engineering Education Conference (EDUCON). Presented at the 2020 IEEE Global Engineering Education Conference (EDUCON), IEEE, Porto, Portugal, pp. 1563–1567. https://doi.org/10.1109/EDUCON45650.2020.9125348

Kourtesis, P., Collina, S., Doumas, L.A.A., MacPherson, S.E., 2019. Technological Competence Is a Pre-condition for Effective Implementation of Virtual Reality Head Mounted Displays in Human Neuroscience: A Technological Review and Meta-Analysis. Front. Hum. Neurosci. 13, 342. https://doi.org/10.3389/fnhum.2019.00342

Leskovsky, R., Kucera, E., Haffner, O., Rosinova, D., 2020. Proposal of Digital Twin Platform Based on 3D Rendering and IIoT Principles Using Virtual / Augmented Reality, in: 2020 Cybernetics & Informatics (K&I). Presented at the 2020 Cybernetics & Informatics (K&I), IEEE, Velke Karlovice, Czech Republic, pp. 1–8. https://doi.org/10.1109/KI48306.2020.9039804

Li, W., Nee, A., Ong, S., 2017. A State-of-the-Art Review of Augmented Reality in Engineering Analysis and Simulation. MTI 1, 17. https://doi.org/10.3390/mti1030017

Li, W.K., Nee, A.Y.C., Ong, S.K., 2018. Mobile augmented reality visualization and collaboration techniques for on-site finite element structural analysis. Int. J. Model. Simul. Sci. Comput. 09, 1840001. https://doi.org/10.1142/S1793962318400019

Lin, J.-R., Cao, J., Zhang, J.-P., van Treeck, C., Frisch, J., 2019. Visualization of indoor thermal environment on mobile devices based on augmented reality and computational fluid dynamics. Automation in Construction 103, 26–40. https://doi.org/10.1016/j.autcon.2019.02.007

Logg, A., Lundholm, C., Nordaas, M., 2020. Finite element simulation of physical systems in augmented reality. Advances in Engineering Software 149, 102902. https://doi.org/10.1016/j.advengsoft.2020.102902

Lydon, G.P., Caranovic, S., Hischier, I., Schlueter, A., 2019. Coupled simulation of thermally active building systems to support a digital twin. Energy and Buildings 202, 109298. https://doi.org/10.1016/j.enbuild.2019.07.015

Pirola, C., Peretti, C., Galli, F., 2020. Immersive virtual crude distillation unit learning experience: The EYE4EDU project. Computers & Chemical Engineering 140, 106973. https://doi.org/10.1016/j.compchemeng.2020.106973

Saredakis, D., Szpak, A., Birckhead, B., Keage, H.A.D., Rizzo, A., Loetscher, T., 2020. Factors Associated With Virtual Reality Sickness in Head-Mounted Displays: A Systematic Review and Meta-Analysis. Front. Hum. Neurosci. 14, 96. https://doi.org/10.3389/fnhum.2020.00096

Shi, H., Ames, J., Randles, A., 2020. Harvis: an interactive virtual reality tool for hemodynamic modification and simulation. Journal of Computational Science 43, 101091. https://doi.org/10.1016/j.jocs.2020.101091

Solmaz, S., Dominguez Alfaro, J.L., Santos, P., Van Puyvelde, P., Van Gerven, T., 2021. A practical development of engineering simulation-assisted educational AR environments. Education for Chemical Engineers 35, 81–93. https://doi.org/10.1016/j.ece.2021.01.007

Solmaz, S., Van Gerven, T., 2021. Automated integration of extract-based CFD results with AR/VR in engineering education for practitioners. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-10621-9

Su, S., Perry, V., Bravo, L., Kase, S., Roy, H., Cox, K., R. Dasari, V., 2020. Virtual and Augmented Reality Applications to Support Data Analysis and Assessment of Science and Engineering. Comput. Sci. Eng. 22, 27–39. https://doi.org/10.1109/MCSE.2020.2971188

Tian, W., Han, X., Zuo, W., Sohn, M.D., 2018. Building energy simulation coupled with CFD for indoor environment: A critical review and recent applications. Energy and Buildings 165, 184–199. https://doi.org/10.1016/j.enbuild.2018.01.046

Tian, Z.F., 2017. Teaching and enhancement of critical thinking skills for undergraduate students in a computational fluid dynamics course. International Journal of Mechanical Engineering Education 45, 76–88. https://doi.org/10.1177/0306419016674133

Van Merriënboer, J.J.G., Kester, L., 2014. The four-component instructional design model: Multimedia principles in environments for complex learning., in: The Cambridge Handbook of Multimedia Learning, 2nd Ed., Cambridge Handbooks in Psychology. Cambridge University Press, New York, NY, US, pp. 104–148. https://doi.org/10.1017/CBO9781139547369.007

Wang, R.-X., Wang, R., Fu, P., Zhang, J.-M., 2020. Portable interactive visualization of large-scale simulations in geotechnical engineering using Unity3D. Advances in Engineering Software 148, 102838. https://doi.org/10.1016/j.advengsoft.2020.102838

Wehinger, G.D., Flaischlen, S., 2020. Studying Computational Fluid Dynamics in a New Dimension with Virtual Reality, in: Computer Aided Chemical Engineering. Elsevier, pp. 2041–2046. https://doi.org/10.1016/B978-0-12-823377-1.50341-4

Zhu, Y., Fukuda, T., Yabuki, N., 2019. Integrating Animated Computational Fluid Dynamics into Mixed Reality for Building-Renovation Design. Technologies 8, 4. https://doi.org/10.3390/technologies8010004