Edwards et al. Visualization in Engineering (2015) 3:4
DOI 10.1186/s40327-015-0018-2

Visualization in Engineering
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

BIM based collaborative and interactive design process using computer game engine for general end-users

Gareth Edwards[1], Haijiang Li[2*] and Bin Wang[2]

Abstract

Background: The emerging Building Information Modelling (BIM) in the Architectural, Engineering and Construction (AEC)/Facility Management (FM) industry promotes life cycle process and collaborative way of working. Currently, many efforts have been contributed for professional integrated design/construction/maintenance process, there are very few practical methods that can enable a professional designer to effectively interact and collaborate with end-users/clients on a functional level.

Method: This paper tries to address the issue via the utilisation of computer game software combined with Building Information Modelling (BIM). Game-engine technology is used due to its intuitive controls, immersive 3D technology and network capabilities that allow for multiple simultaneous users. BIM has been specified due to the growing trend in industry for the adoption of the design method and the 3D nature of the models, which suit a game engine's capabilities.

Results: The prototype system created in this paper is based around a designer creating a structure using BIM and this being transferred into the game engine automatically through a two-way data transferring channel. This model is then used in the game engine across a number of network connected client ends to allow end-users to change/add elements to the design, and those changes will be synchronized back to the original design conducted by the professional designer. The system has been tested for its robustness and functionality against the development requirements, and the results showed promising potential to support more collaborative and interactive design process.

Conclusion: It was concluded that this process of involving the end-user could be very useful in certain circumstances to better elaborate the end user's requirement to design team in real time and in an efficient way.

Keywords: Building Information Modelling (BIM); AEC/FM; Computer game engine; End user; Collaboration

Background

Building Information Modelling (BIM) is a new technology/method emerging in AEC/FM industry. It promotes life cycle process and collaborative way of working. The basic concept is that all information/data are shared/updated/maintained by all of the involved parties guided/governed by the appropriate governance/management model. BIM allows for design of a building/structure to be completed in full 3D and also allows for a design to be modified and updated for all parties so that they can see the changes almost immediately and see if those changes affect their elements of the design. Additionally, BIM enables more specialised parts of the design to be completed by others, other than the traditional main three (architect, structural engineer and service engineer). For example it facilitates conducting energy analysis or quantity surveyors to look at the quantities of materials that may be required based on the elements parametric data. It also aids scheduling of the works, known as 4-Dimensional (4D) design. This schedule can be used to create an animation of the construction project, which may be useful when helping professional designer to understand the full construction process or demonstrate it to a client. BIM can also allow the non-technical parties to understand the

* Correspondence: lih@cardiff.ac.uk
[2]Cardiff School of Engineering, Cardiff University, Queen's Building, The Parade, Cardiff CF24 3AA, Wales
Full list of author information is available at the end of the article

design at the design stage, to see the design in 3D and get a proper visualisation of the whole project, which is not easily possible with the more traditional methods.

Although, in theory BIM does allow for far greater collaboration between many of the parties involved in the design process, the current practices and developments mainly focus on the collaboration between the different elements of the professional design teams. Many efforts have been contributed for professional integrated design / construction / maintenance process, there are very few practical methods that can enable a professional designer to effectively interact and collaborate with end users (functionally). Currently, with either BIM or traditional design practices the client is generally only provided with passive information on a project via the design team preparing information in some form of formal presentation. On a functional level BIM has not yet allowed for a client to contribute to the design phase (Christiansson et al. 2011). By this it is meant that all of the data that is presented to a client is static and has to be transferred into some sort of other form with no real interactive element for the client. An example of this would be that the model might be presented on paper or on a computer to a client but the views and information will have been pre-prepared by one of the design team to present to the client and may not present all of the information that a client wants to see. This also does not allow the client to actively explore the building in its entirety and view what they want to see not what a designer thinks that they want/need to see.

This paper introduces a software system development addressing the technical issue as mentioned above regarding the involvements of end users in BIM based design process via the utilisation of computer game engine combined with BIM (Christiansson et al. 2011; Eastman et al. 2008). A computer game engine is used due to its intuitive controls (easy enough for non-professional end users), immersive 3D technology and network capabilities that allow for multiple simultaneous users. BIM has been specified due to the growing trend in industry for the adoption of the design method and the 3D nature of the models, which suit a game engine's capabilities. The key contribution is to show a way that allows the functional involvement of non-professional end users in the professional design process. The core research is to develop a two-way data transferring channel between the end users 'playing' in game environment (through Web pages or thin client end) and the professional BIM design team. The other supporting developments include the extraction of the necessary information from the professional BIM model (through C# based Revit API development), game environment development, and setting up the underlying computing environment etc. The developed prototype further extends the current BIM implementation to cover non-professional end users. With further improvement and development, e.g., embedded with the appropriate governance model, it could be used to realize a more integrated life cycle based building design/construction/maintenance process to better address the end user's requirements, and hence to improve the efficiency and productivity in the industry.

The rest of the paper is organized as follows. First, the related work regarding computer game engine with BIM applied in AEC/FM domain has been briefly reviewed, including computer game engine software infrastructure and their various applications. A comprehensive software system development process is then explained in detail. The software system architecture is introduced first via a multi-layered diagram, which includes the client end, game engine environment, BIM environment and the data transmission among different elements. The system design and implementation section covers the main hierarchical system design, key classes design and implementation, development framework selection, the implementation for two-way data transferring, extraction of BIM data and other development details. The system deployment and evaluation is given next, where several testing scenarios have been implemented to check the software development errors and system functionalities. Finally, the conclusion and future work are given.

Game engine applied in AEC/FM domain with BIM

Building Information Modelling (BIM) is now approaching to its tipping point (NBIMS 2007; BSI 2010; BuildingSmart 2010; McGraw_Hill_construction 2010) worldwide to revolutionize the entire workflow for AEC industry. Like other countries all over the world, the UK has completed several strategic BIM implementation plans, which are going to require BIM compatibility (level 2 of BIM in Bew & Richards Figure (BuildingSmart 2010)) by the year of 2016. There is a drive from the current UK AEC industry to implement the comprehensive and advanced BIM practice to fulfil the government requirement. While computer game engines have been developed over the years from simple games such as the first interactive computer game OXO (Obituary 2011) to fully immersive 3D environments. Games are highly profitable, thus developers are willing to invest heavily in resources for games development. This gives rise to a highly developed product. The net result is that games represent a pinnacle that other simulation and virtualisation software struggle to equal (Stuart 2011). All of this leads to very powerful tools that can be used for various purposes.

One of the situations that computer game engine technology is being applied to is that of simulating evacuation of the population of a building in a situation such as a fire (Rüppel & Schatz 2011). The models used in some of these simulations are based on 3D information

taken from BIM applications (Yan et al. 2011). Some of the models use agent based systems where the computer based players are programmed using parameters gained from research and the computer simulates the virtual players attempting to evacuate the building (Rüppel & Schatz 2011). This has been extended and changed from being simply a fully computer driven simulation to one utilising the interactivity of the game engine by human subjects to identify how they react within the simulation and what decisions they make in the stressful situation of a fire. Rüppel and Schatz (2011) took this simulation further introducing stimuli such as smoke machines and 3D optical effect screens etc. to try to bring a new dimension to the game.

Another way in which computer game engines have been put to use when combined with BIM is that of providing an interactive visualisation of a structure. This has been done in (Yan et al. 2011) using a combination of the Microsoft® XNA™ Framework as the game engine and Autodesk® Revit® Architecture as the BIM design application. The system allows users to go on a 'walk-through' from one room to another or simply explore the building freely as they wish. This allows for a far more intuitive method of exploring a design before it is built or whilst the building is under construction when compared to the static 2D drawings produced by most CAD based systems. The system also makes full use of the parametric property aspect of the BIM design process by using the properties that indicate if rooms are linked by a door etc. to allow for path finding in the building. A further use that game engines appear to suit is one of visualising a simulation in a graphical way. They suit themselves to this situation well due to their real-time, high quality 3D graphics capabilities and also their ability to cope with 3D geometric data.

El Nimr and Mohamed (2011) used two different scenarios and two different game engines to achieve visualisations of the simulations that they were running. The first example was that of a system designed to simulate the bidding process for attaining a project from tender. It used a map on which the simulation created jobs that players could then bid for. The second example used a real-life construction scenario and a system was developed to visualise the construction process on a site where parts of the structure were to be built off-site. These were brought to a yard on site for further assembly into modules that were then finally constructed into the finished structure. This system simulated the whole construction process and visualised both the site, enabling the contractors to visualise where cranes could fit on to the site during the construction, and also the yard where the individual parts were to be assembled into modules. Both of the scenarios demonstrated the merit of having a graphical representation of the simulation, which allowed for the developers and also the users of the systems to see what was going on and where that fitted into the project.

Using a computer game engine as part of a design review system may not seem the obvious choice. However, due to the nature of the game engine providing networking features that enable real-time collaboration and real-time 3D graphics for real-time visualisation of a building, a game engine could perform this task well. Currently many of the design review systems that are available provide insufficient allowance for the inclusion of all of the stakeholders who may sensibly contribute to this process. Current methods generally are conducted using peer review. An example of this is the use of a light box to overlay drawings to facilitate discrepancies to be identified between different elements of the design. Due to this sort of process becoming tedious it is found that the process is quite often not conducted efficiently and in some cases is not conducted at all (Shiratuddin & Thabet 2011). A prototype has been created by (Shiratuddin & Thabet 2011), where a game engine has been used to create a game where multiple users (members of the design team) can review a design together and interact with the design. They can see details about the design and do basic 3D editing. It is also possible to communicate with one another within the game and leave notes on the design. The authors envisaged that this sort of design review system will help to improve the quality of the design review that it is undertaken and make it a simpler, easier and more satisfying process.

Another area that game engines have been used for in AEC is that of teaching students about construction site safety (Lin et al. 2011; Dickinson et al. 2011). In work by Lin et al., a game was created that simulates a construction site. The user walks around the site looking for errors or issues with health and safety. When an issue is identified the user selects the issue and then chooses what is wrong with the situation using the games interface. It was noted that, from the students who were test subjects in the work, the system made the learning process more interactive and enjoyable and that it helped them to memorise the health and safety issues. In addition, such virtual environment can enable end users to learn various design skills (Sun et al. 2014; GU et al. 2009). Sun et al. developed a 3D virtual space towards a synchronous distributed design meeting system to allow end-users to sketch or make annotations and have discussions as well as add viewpoints to designs. Gu and Nakapan discussed the benefits and shortcomings of virtual worlds for collaborative design learning and education.

In short, computer game engine has becoming much more advanced and mature (than it was before) to be utilized for AEC/FM industry. In its essence, BIM is life cycle based collaborative process, which would be much more effective if end user gets involved in an efficient

manner right from the beginning stage through to building's demolition. Due to the prompt and direct involvements of end user/client (on a functional level), with good governance model, the way of working can further facilitate the prompt response coming from professional designers, and to some extent it can also mitigate the suffer for designer to repeatedly correct errors due to the lack of effective communication between the end user/client and designer. The development to be introduced in this paper would serve as a good add-on to other developments focusing more on professional teams (Eastman et al. 2008; Gu et al. 2010; Hassanien Serror et al. 2008).

Methods

System design and implementation

The use of game engines provides new and exciting opportunities for technologies such as BIM. However, this sort of utilisation is still in its infancy. Currently BIM does not provide a method to involve the end-users of the structures and promote collaboration between them and the design team. This paper proposes to address this issue by trying to promote collaboration between the design teams and the end-users. The new developed system has to be tested for its robustness and functionality against the development requirements, and the results showed promising potential to support more collaborative and interactive design process between professionals and end-users.

In order to construct a BIM based collaborative design environment, the following steps have been identified and concluded to guide the design of the system, and the multi-layered system architecture is showed in Figure 1.

- Identification of the program suite the designer will use to create their designs
- Establish how the designs will be conveyed to the end-user
- Ascertain how to enable the user to have a simple and intuitive process to modify designs
- Determination of a method to transfer the information relating to these changes to the designer
- Identification of the modules, data sources and executables that are required to support the task.
- Plan the order and structure of the implementation.
- Plan the methods used for evaluation and testing of the system.

The multi-layered system architecture comprises four inter-connected layers (Wang et al. 2014):

- BIM environment: Autodesk Revit can be regarded as a comprehensive building information provider to work with game technology to build an adaptable immersive serious game
- The data transmission element: Revit plug-in contains the database and FBX format converter. It works as 'micro' web-server to generate semantic and geometric data and store it in the database to build a two-way and dynamic information flow for collaborative and interactive design.
- Game Engine environment: Unity game includes the Unity server and client game. A unity server component connects Unity clients into an instance and feeds those clients the available data which is created in the data component. Unity clients are the interface that helps immerse the end-user into the virtual environment which is generated by the Unity server (controlled by the designer).
- The client end: End-users and designers can work together in different platforms supported by Unity game engine with appropriate input and output devices, i.e. Windows or Mac operating systems that use high-resolution monitors with keyboard and mouse; 3D stereoscopic projector with Razer Hydra

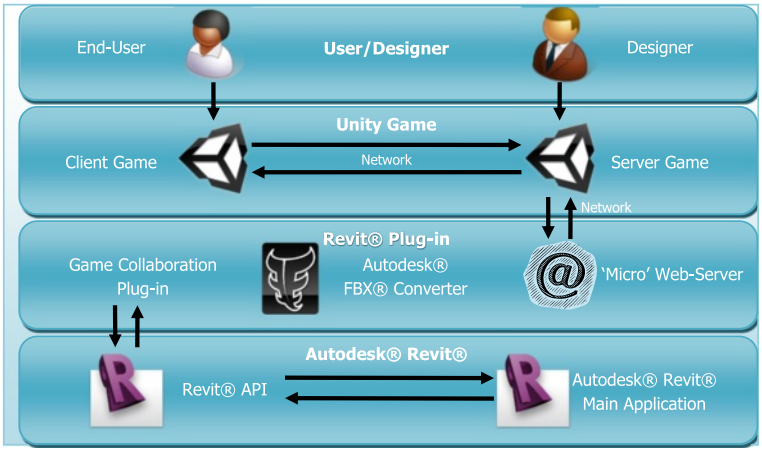


Figure 1 system overall architecture.

joystick; Hand Mounted Display (HMD) with Microsoft Kinect Sensor; Mobile platform using iOS or Android with touch screen and built-in camera; or Web-based environment that allows users to connect to the server through their web browser.

Implementation framework selection

The major consideration in determining the choice of system development framework was that of compatibility of many of the BIM standards that have been laid down. A further significant consideration was one of third party support with software development kits. Taking these factors into account, Autodesk® Revit® Architecture was selected being fully compatible with BIM standards and having excellent third party developer support (AEC (UK) BIM Standard for Autodesk Revit (2010)). It should be noted that the system would also function with Autodesk® Revit® Structure and Autodesk® Revit® MEP (Revit 2012 API 2011) (when working with different professionals). The method of enabling the end-user/client to be presented with and modify information contained within the BIM model must meet the following criteria. The method must be able to display the 3D nature of the model and provide a simple, intuitive interface. This method must also provide some way to facilitate the end-user/client to collaborate with other stakeholders. A suitable system that has been decided upon is that of a game engine. A game engine provides powerful real-time 3D graphics, networking features that allow for collaboration and a simple and intuitive interface (Moore 2011; Petri et al. 2014). Unity 3D from Unity Technologies has been chosen as the platform to interact between the professional designer and the end user/client. This particular engine has been chosen due to its simple object orientated and editor based design system. It also provides many features that other engines do not such as the ability to create executables that will run on both Microsoft® Windows® and Apple® Mac®. It also provides the ability to create 'Web player' versions of any game produced, which will be a useful tool when providing a method of simple delivery to the end user as it would mean that the end user could use their own web browsers without having to download and install the game. Currently to use the Unity web player a plug-in must be installed into the web browser being used. This may be negated by using the support for creating an Adobe® Flash® Player object in the 3.5 version of Unity that has been released recently (at the time of writing). Adobe® Flash® Player is a common web technology is utilised on almost all modern websites.

Key classes design

All the classes developed in the system help to deliver the collaborative design environment. Figure 2 shows the work flow that occurs within the game.

When the server starts up, the first procedure to implement is to create an instance in the database to store this information via the menu system. Next it waits for the building information to load. The loading time varies depending on the amount of information in the BIM model. The internal process of loading data is to convert Revit building format to FBX format. Then the FBX model is loaded into the Unity server's memory and converted into our custom format, which buffers at the network layer for incoming clients to load. The client first needs to connect to the server by entering the server IP address, and then receive the building information. The two highlighted grey sections are specific for the client version of the game and their counterparts are specifically for the server version of the game. The end users can control the avatar and toggle main menu for different tasks in the server and clients.

The entire system development follows a sound software engineering approach, specific activities such as applying metrics and identifying characterisations such as design requirements (aforementioned) and the level of details are important in that activity. In relation to the object oriented software development approach used, the overall system class hierarchy to constitute menu structure and game for collaborative and interactive design is shown in Figures 3 and 4, and the following contents show the key classes design details.

All of the classes except 'OBJModel' are inherited from 'MonoBehaviour', which is Unity's base class.

Main menu

The 'MainMenu' class which is a single script attached to the camera object in the 'MainMenu' game scene in the Unity game. As can be seen (in Figure 4) it has four variables of which three are public to allow access to them in the Unity editor. These three public variables are a skin for the GUI components to use, a backdrop for the menu and the title or name of the game to display on the menu. These three variables are common on all of the menu scenes as is the fourth variable 'isLoading', which is private and is used for determining if the game is changing scenes between a menu scene or the game scene. Finally it can be seen that there is a subroutine 'OnGUI()' which is a Unity built in subroutine that gets called by the game and is used to create the GUI elements and in this case implement the logic for the buttons of the main menu.

Start server

The 'StartServer' class, which is attached to the camera object in the 'StartServer' scene. This being a menu has the same variables as the main menu and they will not be described again. It also has some other variables, which are all for the various settings used for the server.

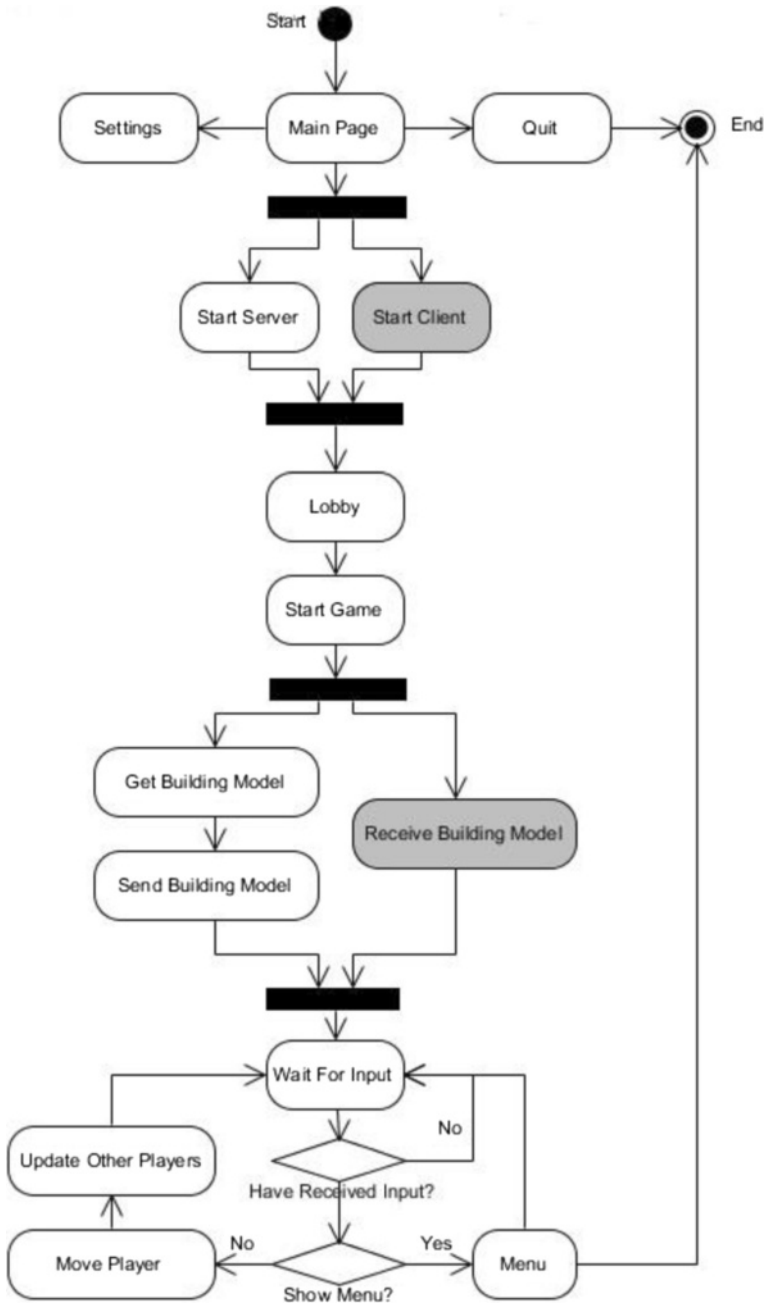


Figure 2 Activity UML model for game flow.

All of the variables are all public other than the player number as they need to be accessed by the Unity editor to allow the default settings to be specified. The 'Update ()' subroutine, which is built in Unity subroutine that executes once per frame render in the game. It is implemented here to allow for capturing of input keys namely the 'escape' key to get back to the main menu. The 'PlayerName' is included to set a default player preference as part of the network connection starting process. This starting of the game server is initiated in the 'OnGUI()' subroutine along with the other functionality for the controls including moving to the lobby scene.

Connect to server

The 'JoinServer' class is attached to the main camera in the game scene 'JoinServer'. It also again implements all of the previous standard menu variables and the 'OnGUI ()' subroutine. Other than those variables and subroutines it implements two private variables that are used to hold the value of two text input fields for use as settings

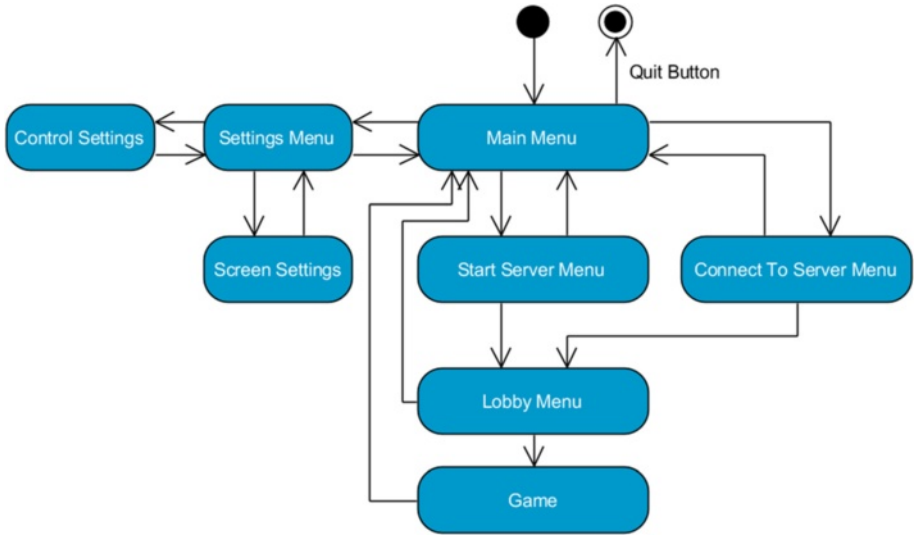


Figure 3 Menu structure of the game.

when trying to connect to a server version of the game. The three other subroutines that are implemented are Unity built in subroutines that as their names suggest deal with various situations of network connectivity and are, in this case, only used to send messages to the debug log. Again the 'OnGUI()' subroutine has been used to implement the functionality of the controls including initiating the connection with the server version of the game and changing to the lobby scene.

Lobby menu

The 'Lobby' class is attached to the main camera object in the 'Lobby' scene. It implements all of the menu variables and the 'OnGUI()' subroutine. Of the four other variables that are implemented two are used for allocating each player (client) a unique number and the other two are used for allocating the settings that the game server will use to connect the building model server (Revit® plug-in). The 'Update()' subroutine is implemented to capture the 'escape' key input to return to the main menu. The 'Awake ()' subroutine is another Unity built in subroutine that initiates before any of the rest of the class starts, unlike the 'Start()' subroutine where some of the class is started before the function runs. It is used to run what is known as a Remote Procedure Call (RPC), which is essentially a method of allowing the same function or subroutine to be initiated on all connect games, in different locations, at the same time. It runs the RPC subroutine 'GetPlayerNumber()', which in turn runs another RPC, namely 'ReturnPlayerNumber' from the clients so that they each request a player number from the server. The functionality of the controls, which are only shown on the game server not the clients, is implemented in the 'OnGUI()' subroutine including executing the 'LaunchGame()' RPC on all clients that makes the server and all clients change scenes to the 'Game' scene.

Player in game

The 'MovePlanar' class is attached to the 'Player' prefab, which is an object that can be instantiated (created) across the network. There are five public variables to allow for the editor to set the values of them before the game is complied. These include speed factor which are used in setting the speed of movement of the player, a rotational sensitivity, which again is essentially a speed factor and the Skin to use for the GUI. There are also a number of Boolean type variables that are used to enable and disable motion and menus depending on inputs such as the 'escape' key, which is captured in the 'Update()' subroutine along with the input and implementation of the movement algorithm. There is also the 'Position', which is of Vector3 type (3D vector), is used for maintaining the position of the player across the network on the clients games. The 'Awake()' subroutine is used to get the RigidBody component of the 'Player' prefab to use for enabling and disabling gravity. The 'OnGUI()' is again used for implementing a menu that is only displayed after the 'escape' key has been pressed with functionality such as exiting the game or enabling motion in the Y axis plane, whilst disabling the gravity option of the RigidBody component. Two other Unity built in subroutines are also used in this class. 'OnSerializeNetworkView()' is used for sending information about the state of gravity and position of the 'Player' prefab across the network to the other players (clients). The second subroutine 'OnNetworkInstantiate()' is used to ensure that the camera is correctly configured to be the one that is a child of the 'Player' prefab.

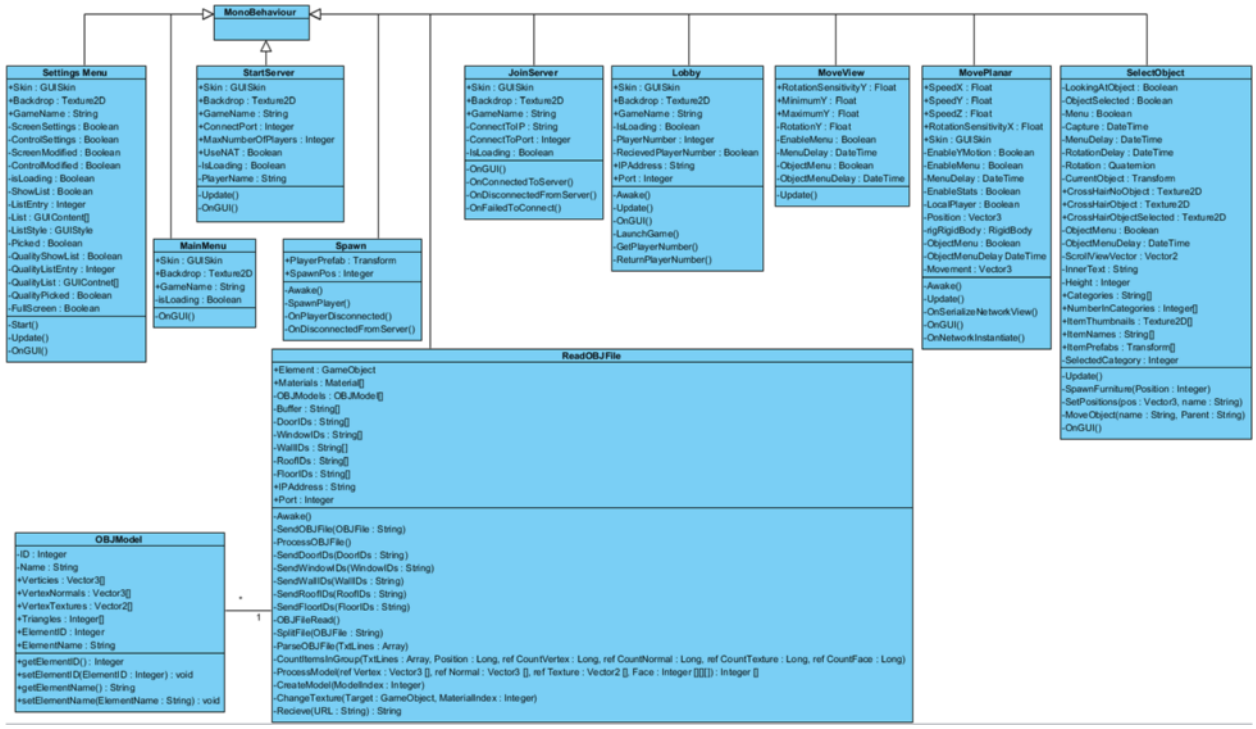


Figure 4 Overall system class hierarchy.

The 'MoveView' class, shown in the class hierarchy is attached to the camera object which is a child of the 'Player' prefab. The first three public variables are used to set the parameters for the rotational movement in the XY plane. The private 'RotationY' is used to hold the previous value of rotation to be used to rotate the camera. This is not sent across the network as the camera object is not seen by the clients and hence does not require moving on the client side. The last four variables are used to disable and enable the motion of the camera depending on which menus are active. All of this functionality is implemented in the 'Update()' subroutine.

The 'Spawn' class is attached to 'Spawn' game object. This class initiates the 'spawning' (creation) of all of the 'Player' prefabs (one for each connected client and one for the server). The public variable 'PlayerPrefab' is used to store a reference to the 'Player' prefab, which is made using the Unity editor hence the variable being public not private. The 'SpawnPos' is public due to the 'Lobby' class accessing it to give it the player number so that the spawn script spawns all of the players away from one another using the player number to give each a unique spawn position. The 'Awake()' subroutine is used to ensure that the game is connected to/is a server and then initiates the 'SpawnPlayer()' subroutine, which is not a Unity built in subroutine. The 'SpawnPlayer()' then instantiates the current players 'Player' object for them to use as a protagonist in the game. The 'OnPlayerDisconnected()' and 'OnDisconnectedFromServer()' are Unity built in subroutines and are used to make sure the various players are destroyed correctly if a client leaves or if the server disconnects the whole game.

Design in game

The 'SelectObject' class is attached to the 'Camera' game object like the 'MoveView' class. The top two Boolean types are used to tell all of the subroutines within this class whether an object is within range and being looked at by the camera or whether an object within range of the camera has been selected to move it around. 'Menu' and 'MenuDelay' are used as before to indicate if the menu is being displayed and to stop the 'escape' key initiating the menu appearing an disappearing if the 'escape' key is held down. 'Capture', 'RotationDelay', 'Rotation' and 'CurrentObject' are all used in the movement of an object that has been selected by the player, which is implemented in the 'Update()' subroutine. 'ObjectMenu', 'ObjectMenuDelay', 'ScrollViewVector', 'InnerText', 'Height' and 'SelectedCategory' are all used to implement the scroll view component implemented as another menu that is used to create objects selected from the menu. The objects contained within this menu are implemented using the public variables 'Categories', 'NumberInCategories', 'ItemThumbnails', 'ItemNames' and

'ItemPrefabs'. These allow for the specification of categories using the 'Categories' variable along with the number of items contained in that category using 'NumberInCategories'. The items three details, their names, thumbnail images and references to their prefabs, are then placed in the same order as the categories in the other three variables. All of this scroll view is implemented in the 'OnGUI()' function up until an item is selected by pressing a button containing the thumbnail of an object. This then runs the 'SpawnFurniture()' subroutine, which creates an instance of the selected prefab in front of the player. This in turn runs the 'SetPositons()' RPC, which moves the object on the other players games to its start position. The 'MoveObject()' RPC is supposed to allow for movement of the objects by sending the name of the object to be moved and also the name of parent (player) that will be moving the object. It then utilises these names to find the objects and tries to move the object with the parent object (player).

Data communication between BIM environment and client end

The realization of BIM data extraction for computer game engine is through using C# based Revit API development. Figure 5 shows the class for BIM data extraction and the extraction process. The 'StartGameEngineViewer' class is used to deal with all of the functionality contained in the plug-in. It is initialised by the button on the ribbon bar created in the 'AddInPanel' class. The first function that is called in the 'StartGameEngineViewer' class is the built in Revit API function 'Execute()'. Therefore, the 'Execute()' function is public because the 'AddInPanel' class needs to access it. The global variables in this class are used mainly due to there being a background worker that runs on a separate thread, which still needs access to some of the data. The 'OBJFile' variable is used to hold a full copy of the '.obj' file for the current building model. The 'uidoc' variable is used to hold a handle to the current document in Revit®. This is so that all of the functions and the background worker can access the document. The 'listener' is object that makes up the 'mini' web-server along with the background worker 'bgWorker'. The 'Domain' and 'PORT' variables are used to hold variables for the settings for the listener. The 'CloseListener' is used so that the main thread can make the background worker thread terminate. The 'AppName' variable is used to hold the name of the plug-in as it may be used in the requests to the 'listener' object and will need to be identified in a requested URL and removed.

Figure 6 shows the 'ReadBJFile' and 'OBJModel' classes, which are attached to the 'BuildingModel' game object. The class 'ReadOBJFile' allows for the creation of the building by getting the geometric and some parametric data from the plug-in in Revit® using '.obj' (Wavefront 3D) files. The class 'OBJModel' is used to create an array of the individual objects contained in the building model within the 'ReadOBJFile' class. The 'OBJModel' class essentially is just a container for the variables it needs to hold, which are the ID number and name of the element. It has implemented get and set functions for both of these variables. It also holds an array of 3D vectors (Vector3) for the vertices and the normal's to the faces, which specify the direction in which a face is visible from. 'OBJModel' also holds an array of 2D vectors used to hold the vertex textures or UVs, which are used to position a texture on a face of the model correctly. Finally there is an array of integers that is used to hold a list of indexes to the vertices, normals and UVs used to define each triangle that makes up the mesh of the object.

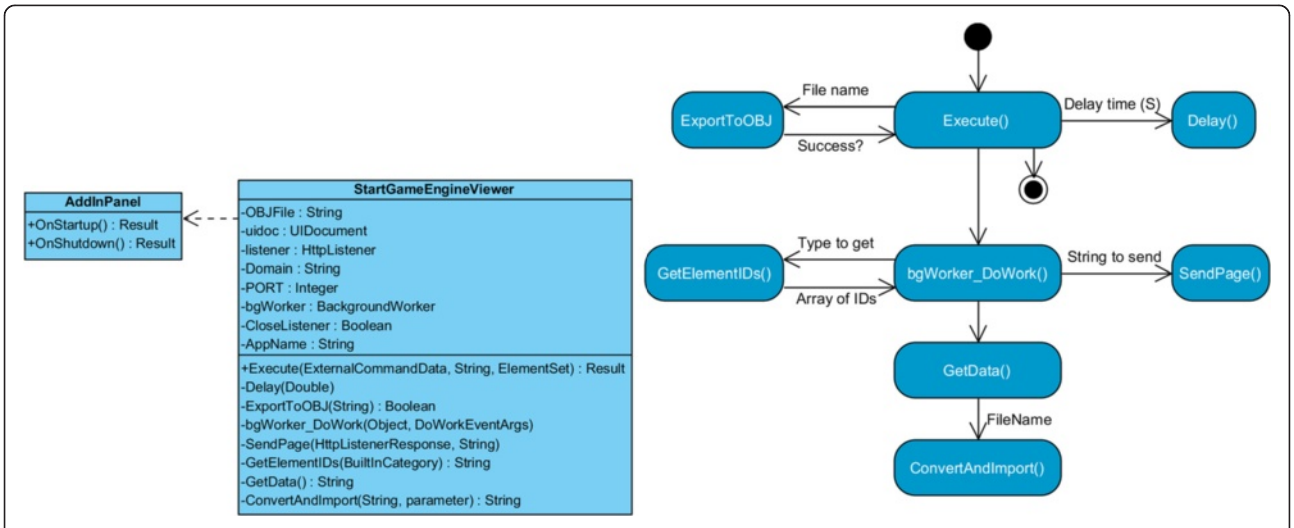


Figure 5 Plugin class and data extraction process.

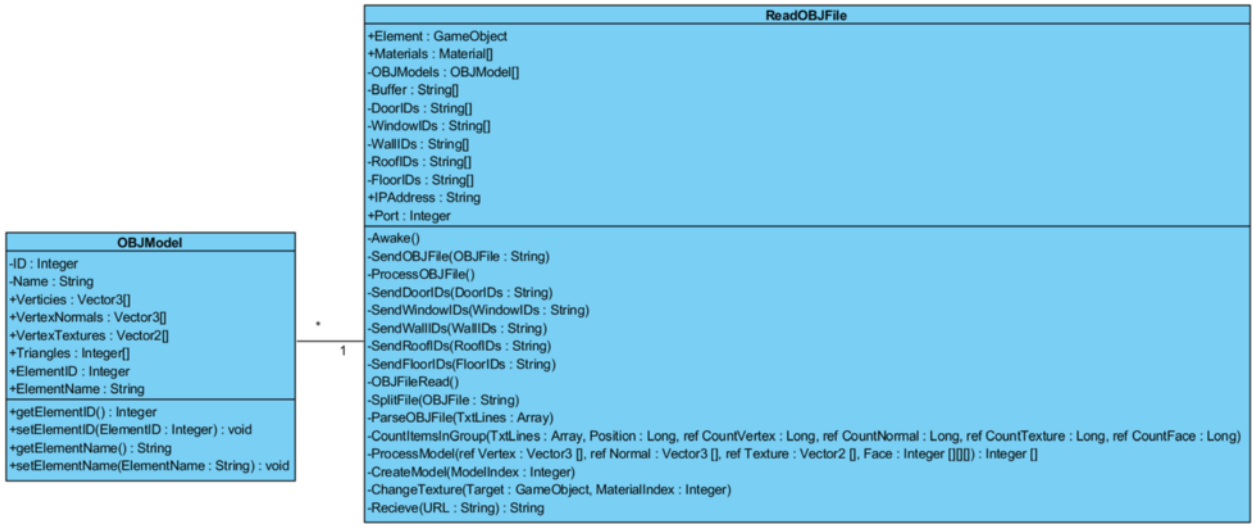


Figure 6 OBJModel and ReadOBJFile classes.

The 'ReadOBJFile' class contains many different variables, functions and subroutines. The first two variables are public so that they can be accessed through the Unity editor. 'Element' is used to hold a reference to the empty prefab only containing the correct components such as a RigidBody to allow for physics interaction this facilitates the building mesh to be built. An instance of 'Element' is used for every element contained in the building. 'Materials' is used to hold an array of type Material that are placed in the array in the editor and are used for applying textures to the various elements of the building. The next private variable 'OBJModels' is an array of OBJModel classes and is used as described to hold the data for the elements ready for creation in the game. Buffer is used to act as a temporary storage location for elements of the '.obj' file that is to be sent to the other players from the server version of the game. The next five private variables are used to hold arrays of strings that will contain element IDs. These are used to apply the correct textures to those five types of elements and to create colliders on only the elements that are not doors so that free movement around the building is possible. The final two variables are public and are used to set where the web client tries to access the building model information ('.obj' file and element IDs).

In 'ReadOBJFile' the only built in function or subroutine used is that of the 'Awake()' subroutine, which is used to initiate other subroutines. The sequence when obtaining a model from the building model 'mini' webserver is as shown in Figure 7. The names of many of the functions and subroutines should be self-explanatory except for the 'CountItemsInGroup()', which is used to count the next set of vertices, vertex normals, UVs and faces of the 3D model, where each set is an element of the building such as a door. Another function that may not be immediately obvious is the 'ProcessModel()' function that is used for transforming the layout of the information in the '.obj' file to the format that Unity uses.

Results and discussion

System deployment and evaluation

In order to fulfil the system development requirements, the following criteria have been concluded for the finished system to be assessed against. These criteria have been evaluated through all stages of the implementation and testing (developer testing) but have mainly been checked using the two test scenarios described later (functionality testing).

The section 4.1 and 4.2 demonstrate how to meet criteria about information communication:

- The system must be able to import a building model from Autodesk® Revit® into the Unity 3D based game.
- Communication between the Autodesk® Revit® plug-in and the Unity 3D game must be possible.
- Properties of elements contained in the Revit® model must be able to be communicated to the Unity 3D game such as parametric properties or element type properties.

The section 4.3 illustrates how to meet functional criteria for end users:

- All of the aforementioned items of data (building model, properties, etc.) must be able to be forwarded to clients.

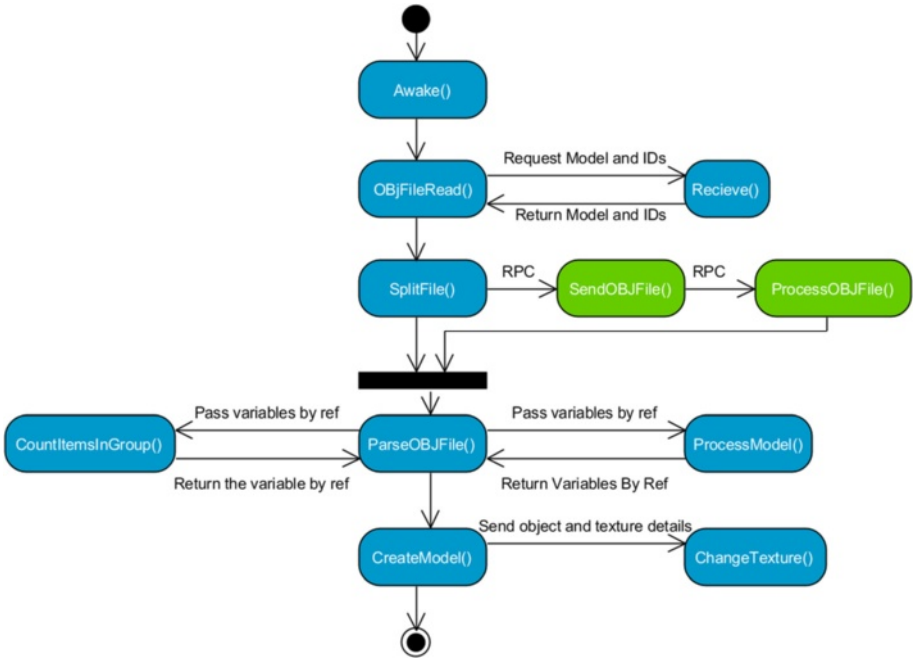


Figure 7 Data transferring process.

- Models of furniture and other items must be able to be placed by an end user on a client version of the game or by the designer into the building model.
- Where client (end users) versions of the game place a model in their game the information about that placed object must be sent to the server version of the game.
- The designer must be able to transfer information about the objects placed by the clients back to Autodesk® Revit®. The plug-in will then load this information into the building model.

The following sections demonstrate the collaborative design system prototype working as intended, when assessed against the evaluation criteria. All of these tests throughout the testing section were carried out in a networked computing environment running Microsoft® Windows® 7 with Autodesk® Revit® 2012 and Unity 3.4 installed.

Setting up the linking between BIM modelling & game engine

The plug-in appears in the ribbon bar of Autodesk® Revit® as a button that will initiate the exporting of the model and set-up the building model server. This plug-in is shown in Figure 8 in the 'Add-Ins' Section of the Revit® Ribbon Bar.

The form shown in Figure 8 at the bottom, allows for the changing of settings for the 'mini' web-server. These settings include the Internet Protocol (IP) address that the server listens on and also the Port that the server listens to. Finally there is a check box to indicate if the game is to be run on the local machine. This option launches the game in its current location on the local machine automatically after the 'mini' web-server has started. The plug-in also has another form that is displayed, which is shown in Figure 8 on the left of the diagram. This form is displayed due to the nature of the Revit® Application Programming Interface (API), which is a transaction based system. Systems of this nature do not allow the game to retrieve the data required from Revit® whenever it wants to since 'mini' web-server will be started. Once the transaction has completed, Revit® closes the plug-in and the 'mini' web-server with it. To overcome this, the form shown in Figure 8 on the left is displayed to keep the transaction running. This prevents the designer from closing the form but enables the game to obtain its information. Once the game has the information required, the form can be closed along with the 'mini' web-server so that a designer may continue to use Revit®.

Setting up the game server and game client

Figure 9 shows the process of setting up a server game and a client game once the plug-in has been started as shown in Figure 8.

Testing scenarios

Testing of the system has been conducted in different ways at different points in the implementation life cycle. Developer testing has been carried out continuously through the implementation of the project. Both component

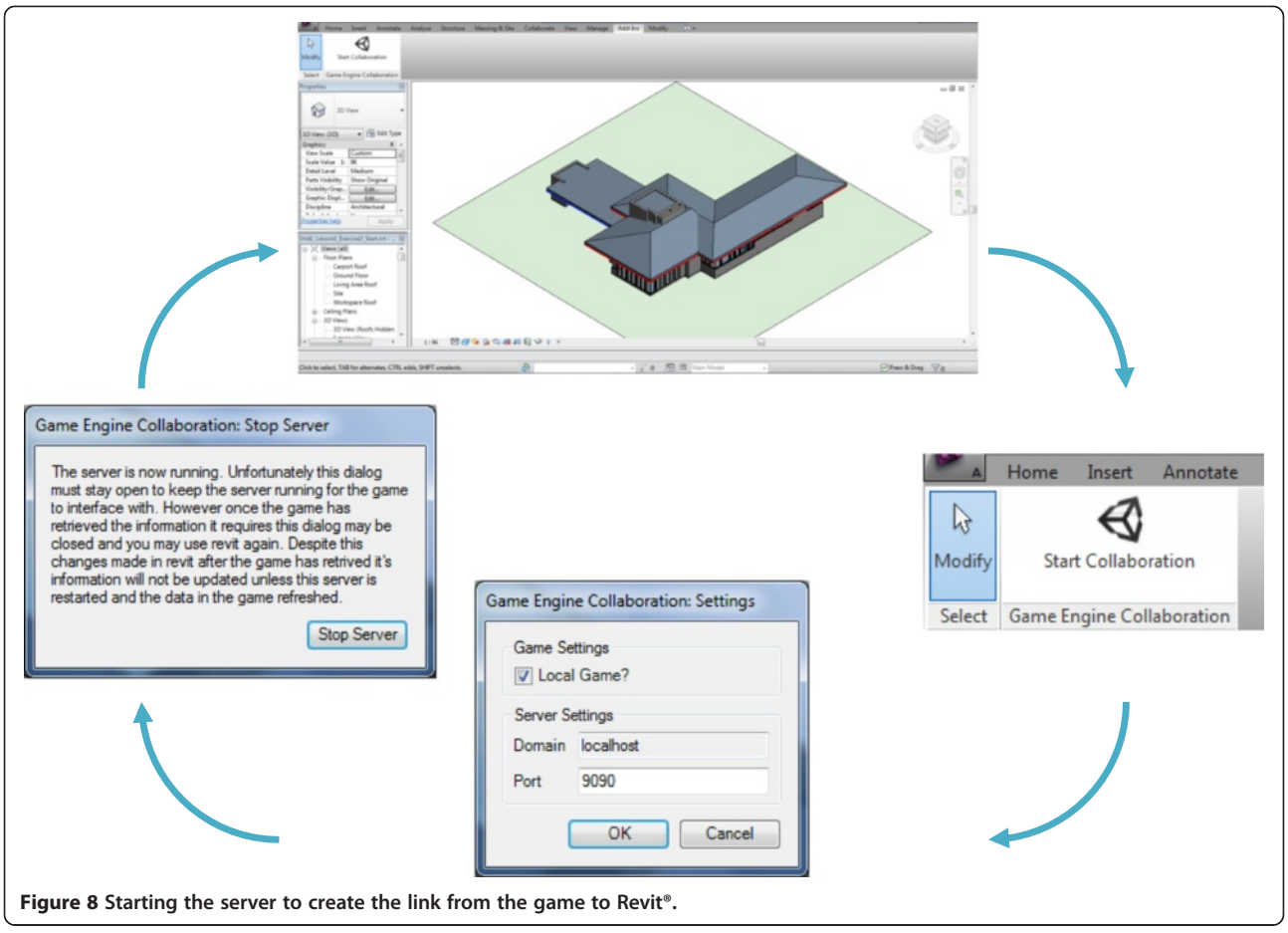


Figure 8 Starting the server to create the link from the game to Revit®.

tests and full system tests were undertaken so that the code could be debugged and corrected during the course of the implementation. Once all of the developer testing and implementation has been completed the system was then functionally tested with the two scenarios described below. The first scenario utilised a block of flats created by the designer into which the clients placed objects, which were transferred back to Autodesk® Revit®. This process is illustrated in Figure 10.

The second scenario employed a single storey house design created by the designer. Within this house one of the rooms was decided to be a kitchen. The user then placed the kitchen of their choosing into this space. The BIM models and Game environment for this are shown in Figure 11, and the similar sequence modelling (to the one showed in Figure 10) has been used.

The second example shows one new build housing estate including many properties, it allows different clients (end users) allocated to each of the properties to place their respective kitchens. The two previous examples only use the example of furniture as objects to be placed. This is not the only use of the system. It could be used to place machinery in a factory before construction to give a lay out, or to place other plant on a construction site. However, in this paper furniture has been used to demonstrate the system.

Throughout the implementation each element of the plug-in and game has been continuously tested. For example every one of the controls has been checked to see if they performed and completed the expected operations. During the development process, any bugs that were encountered were rectified before the next element was implemented. The main way in which the system as a whole was tested was the use of the two scenarios laid out previously (Figures 9 and 10). These were carried out following the procedure seen in the sequence diagrams previously. For both scenarios it was necessary to create test buildings in Revit. As showed in Figures 10 and 11, it can be seen that the two test buildings in Revit® on the left and then imported into the game on the right.

For the next part of these tests the game was started with a client connected, to place their required objects within the building. The placing of objects in the client, within the block of flats model, can be seen in Figure 12. To achieve this, the server version of the game has successfully imported its own copy of the geometry of the model from the plug-in and then sent this to the client for both the server version and the client version

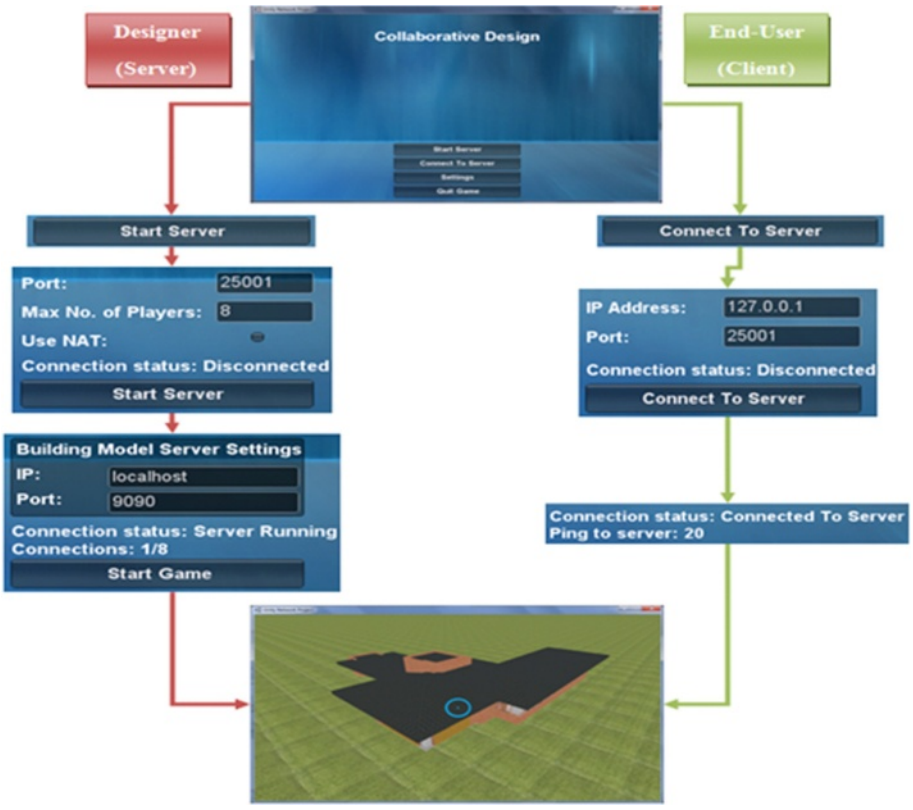


Figure 9 Process to set up a server game and connect a client game.

to build up the 3D model within their respective gaming environments.

In Figure 12 it can be seen that the client has placed the objects where he desired on the game screen, these changes (objects positions and rotations) will then be sent back to the server, which will send the changes further back to BIM design environment. This process is initiated on the server game, which the designer would be in control of.

System evaluations

A total of thirty individuals took part in the test, each using the three different Unity3D clients (including five modes) to finish basic user-centred tasks and completed a pre-experiment and post-experiment questionnaire. The average response was recorded for analysis except for situations where there was a clear bi-polar response when the responses are illustrated individually.

The participant questionnaire responses expanded on the differences between each interface focusing on manoeuvrability, immersive feeling, realistic feeling, quality of model representation and level of interest/excitement (Figure 13). Clearly the greatest difference between the interfaces lied in manoeuvrability. The motion sensor virtual reality (VR) system (i.e. Kinect with HMD) scored a low 2.67, equating to a median between 'Difficult' and 'Very Difficult' on the response scale. There is then a significant jump to public VR system (i.e. 3D Projector with Razer Joystick) at 4.38, landing at the moderate side of 'Difficult' and then another significant jump to the tablet interface which scored 6.67 on the moderate side of 'Easy'. The two desktop interfaces came in as clear preferences above 'Easy' which was noted as 7 on the response scale; the flight mode beating its first person alternative by 0.42 points. It seems that time is needed for general end-users to adapt to the the BIM-VE clients with ad hoc virtual reality devices that are not present in their usual life.

There is a slight difference in results for immersive feeling. The tablet and two desktop interfaces showed almost zero differences in immersive feeling whereas there is a noticeable 1.155 average increase with the two VR interfaces, with the Kinect and HMD interface topping the 3D Projector and Razer Joystick system by 0.25 points. It is clear that clients with high quality virtual reality devices can immerse the participants into the virtual environment although the participant requires more time to become familiar with these client devices. In terms of the realistic feeling and quaility of model representation, the participants tended to give the clients they are familar with in their usual life a higher score.

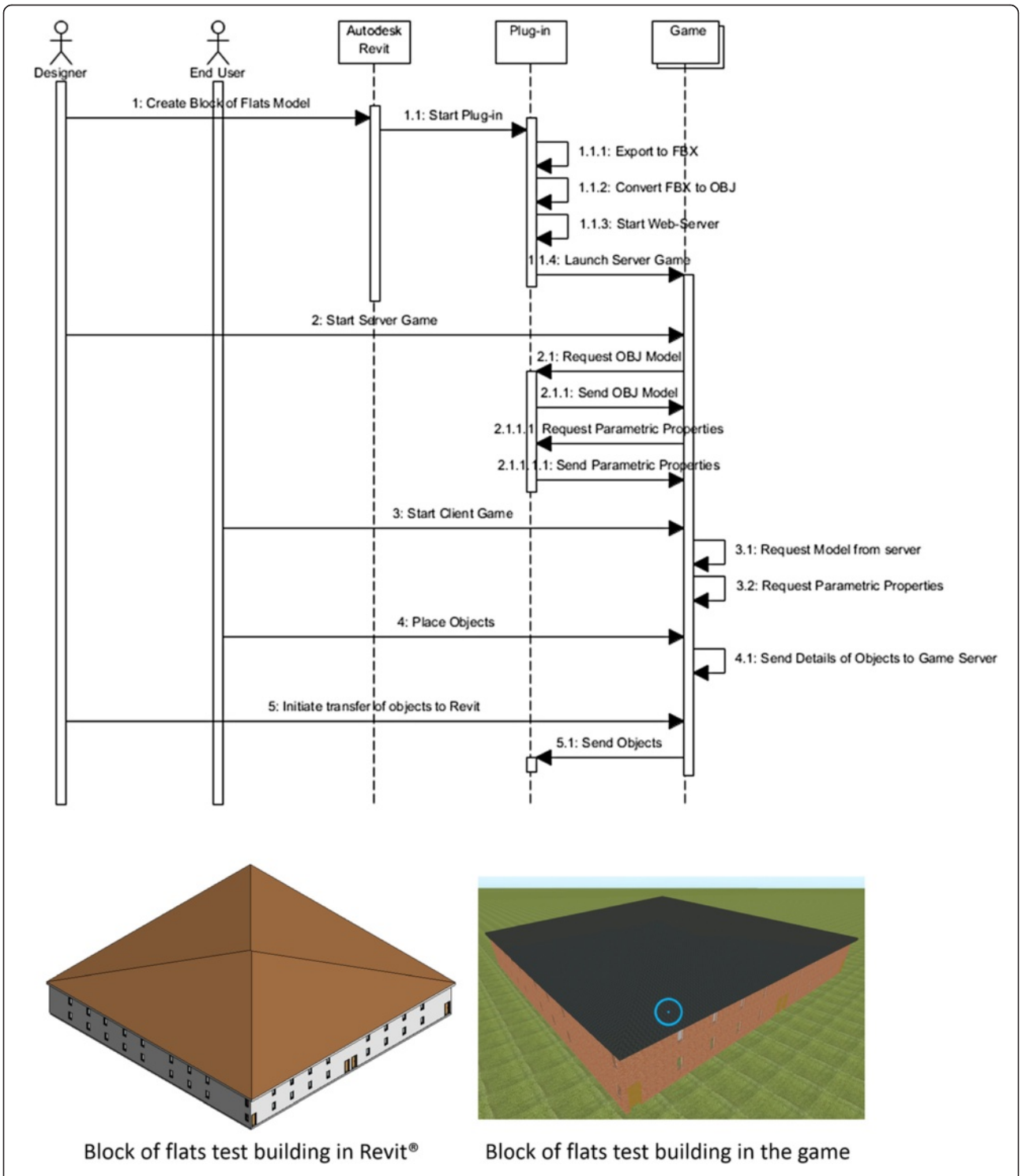
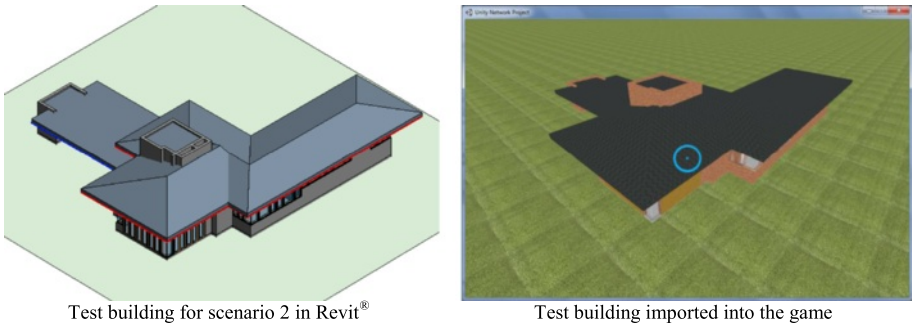


Figure 10 Testing scenario 1.

Finally interest/excitement saw both desktop versions achieving the worst score, with the tablet interface scoring noticeably higher by 0.42 points closely led by the 3D projector with Razer Joystick system, all being rated lower than the Kinect & HMD system. This is because that the more accurate details a VE can map with the real via VR devices, the more immersive effects the end users can feel. It demonstates clearly the VR clients of the BIM-VE hold a huge potential to be popular in the future.



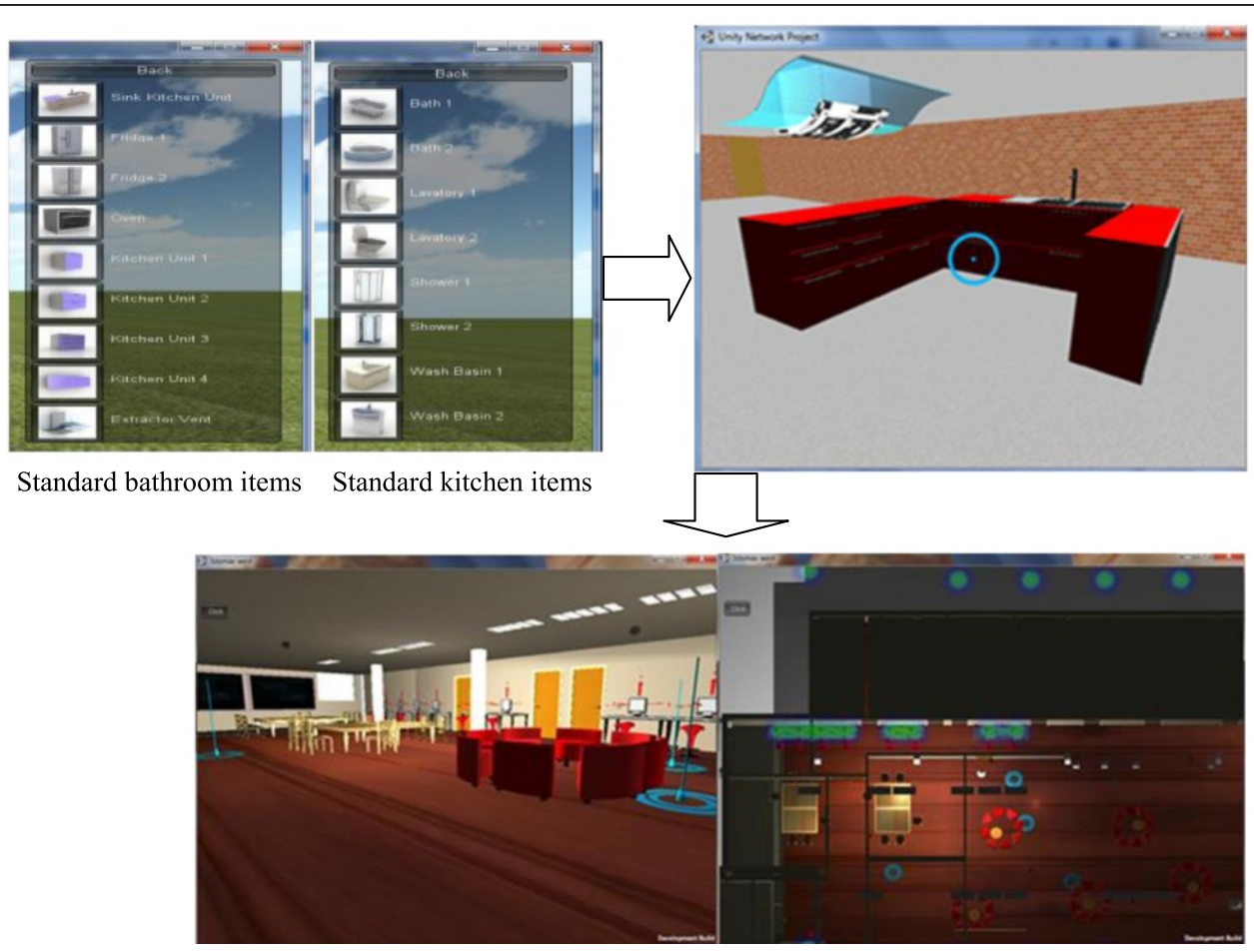
Test building for scenario 2 in Revit®

Test building imported into the game

Figure 11 Testing scenario 2.

Conclusions

The aim of this paper was to increase end-user involvement in the design process in a collaborative manner. The basics of a framework that could be used to achieve this goal have been designed and constructed into a working prototype that allows a user to interact with a design created in a Building Information Modelling (BIM) program, namely Autodesk® Revit® Architecture. This interaction is achieved by the use of a game engine that aims to provide intuitive controls and an immersive environment to allow non-technically trained persons to engage with the building and add to the design. Unlike other studies that have been conducted into the use of BIM and game engine technology, this prototype enables the automatic import of a project that has been designed in Revit®, directly across a network, into the game with no intermediate design steps. The prototype system has also utilised some of the networking



Standard bathroom items Standard kitchen items

Figure 12 Placing object in the client (right) with the server observing (left) to design a room (bottom).

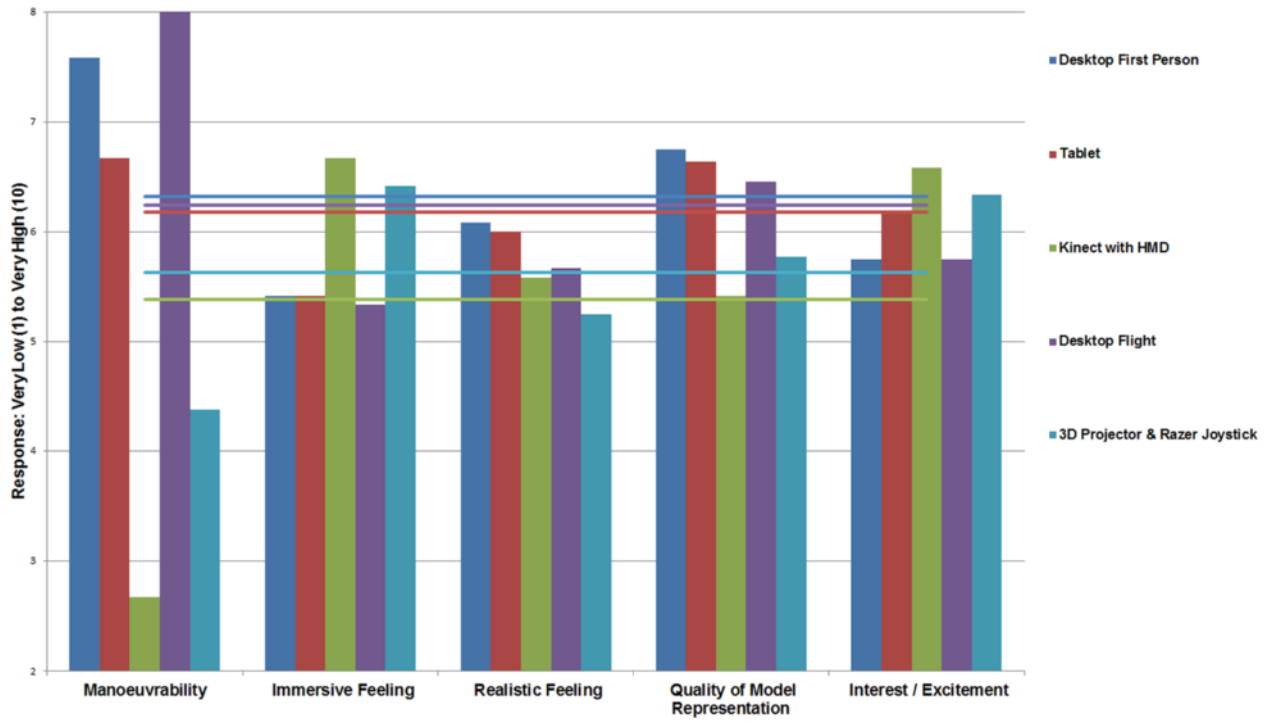


Figure 13 Post-questionnaire response – interface qualities.

potential of the game engine, in this case Unity created by Unity Technologies, allowing for real-time collaboration between persons in the gaming environment. This has been augmented with the ability for a user to place objects within the model to allow for them to adapt part of the structure with items such as furniture. Objects may be placed successfully in the server version of the game, and similarly objects may be placed in the client version of the game. Worthwhile progress has been made towards what could be an exciting and useful technology framework that the author believes could substantially improve end-users inclusion in the design process.

Regarding future work, the most obvious improvements are to resolve the issues that have occurred with the clients updating the positions of the objects that they have created/placed and relaying this same information back to Revit®. Analysis and debugging to resolve this relaying of information may be relatively straight forward but potentially time consuming task. Another aspect for improvement is to enable the placement of walls and doors to allow for the selection of an internal layout. This could be implemented to work in a similar manner to the way that walls and doors can be placed and used in BIM applications such as Autodesk® Revit®. This would mean that walls and doors had parametric properties and for example a wall would use the properties of the rest of the building such as the floor and ceiling heights to work out where it should start and end in the vertical direction. Likewise a door for example would only be able to be placed on a wall rather than just in space.

Another improvement that may significantly improve the openness of the system and enable substantial improvements in other features would be to modify the system to use IFC files. IFC files are an open format that most BIM applications support and they allow for the transfer of BIM models between different packages. If the system was modified to use IFC files, rather than obtaining the 3D geometric and category/parametric data separately, it would facilitate several improvements. The first improvement would be that more of the parametric data could be extracted from the model permitting uses of the system in other areas such as using it for minor modifications of the structural elements of the building. Using IFCs would also allow for direct updating and integration of new elements into the model, which would simplify the process and potentially speed it up (Bogen and East 2011).

An additional improvement that would add a great deal of functionality and capability to the system as a whole would be to create a website to host a web version of the game. This would allow users to be authenticated quickly and taken to their specific project in the game, whilst simultaneously improving the ease of access since it would not have to be installed on their computer. It is possible, dependant on how this feature is implemented, that the user would have to install the Unity web player. Alternatively installation of Unity web player could be

avoided and the widely available web-based Adobe® Flash® player utilised since version 3.5 of Unity makes provision to create an Adobe® Flash® object. Another web-based improvement would be to implement an interface that permits a user to browse a retailers' web-site, within the game, to pick out a particular piece of furniture. The game would then download the associated image for the item and seek basic dimensions such as length, width and height, applying this to a generic model of the type of that object.

One further technical improvement could be undertaken would be to implement Universal Plug and Play (UPnP). This permits programs to negotiate with a fire-wall in a network to allow them access across a network with no user configuration (ISO/IEC 29341–1 2011). This would eliminate many of the issues that the system may encounter due to the networked nature of the game and plug-in. This would be largely applicable to the 'mini' web-server and client rather than the game as Unity's networking features already include support for NAT (Network Address Translation) that can bypass many of these problems (Huston 2004).

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

GE - Jointly came up with the idea of creating the BIM-based Virtual environment with LH. Created and programmed the plug-in for Autodesk Revit. Created and Programmed the Unity based 'game' environment. Wrote sections of this paper. LH - Jointly came up with the idea of creating the BIM-based Virtual environment with GE. Wrote sections of this paper. BW - Wrote sections of this paper and reformatted the paper. All authors read and approved the final manuscript.

Author details

[1]ATKINS, The Hub, 500 Park Avenue, Aztec West, Almondsbury, Bristol BS32 4RZ, UK. [2]Cardiff School of Engineering, Cardiff University, Queen's Building, The Parade, Cardiff CF24 3AA, Wales.

Received: 10 October 2014 Accepted: 22 January 2015
Published online: 12 February 2015

References

AEC (UK) BIM Standard for Autodesk Revit. (2010). *Engineering and Construction industry in the UK*. United Kingdom: AEC (UK) CAD & BIM Standards Site.

Bogen, C, & East, W. (2011). *Using IFC Models for User-Directed Visualization in Congress on Computing in Civil Engineering, Proceedings*. Reston: American Society of Civil Engineers.

BSI. (2010). *Constructing the business case - building information modelling*. London: BSI Corporate.

BuildingSmart. (2010). *Investors Report - Building Information Modeling (BIM)*. London: BSI Corporate.

Christiansson, P, Svidt, K, Pedersen, KB, & Dybro, U. (2011). User participation in the building process. *Journal of Information Technology in Construction, 16*, 309–334.

Dickinson, J, Woodard, P, Canas, R, Ahamed, S, & Lockston, D. (2011). Game-based trench safety education: development and lessons learned. *Journal of Information Technology in Construction (ITcon), 16*(2011), 119–134.

Eastman, C, Teicholz, P, Sacks, R, & Liston, K. (2008). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*. Hoboken, NJ: Wiley.

El Nimr, A, & Mohamed, Y. (2011). Aplication of gaming engines in simulation driven visualization of construction operations. *Journal of Information Technology in Construction (ITcon), 16*(2011), 23–38.

GU, N, Nakapan, W, Williams, A, & Figen Gül, L. (2009). *Evaluating the use of 3D virtual worlds in collabora-tive design learning. In the 13th international conference on Computer Aided Ar-chitectural Design (CAADFutures)*. St Leonards Sydney: Icon. Net Pty Ltd.

Gu, N, Singh, V, London, K, Ljiljana, B, & Taylor, C. (2010). Adopting building information modeling (BIM) as collaboration platform in the design industry. In *The Association for Computer Aided Architectural Design Research in Asia (CAADRIA)*. St Leonards Sydney: Icon.Net Pty Ltd.

Hassanien Serror, M, Inoue, J, Adachi, Y, & Fujino, Y. (2008). Shared computer-aided structural design model for construction industry (infrastructure). *Comput Aided Des, 40*(7), 778–788.

Huston, G. (2004). Anatomy: A Look Inside Network Address Translators. *The Internet Protocol Journal, 7*(3), 2–32.

ISO/IEC 29341–1. (2011). *Information technology – UPnP Device Architecture – Part 1: UPnP Device Architecture Version 1.0. 2011*. Geneva: International Organization for Standardization.

Lin, KY, Son, J, & Rojas, E. (2011). A pilot study of a 3D game environment for construction safety education. *Journal of Information Technology in Construction (ITcon), 16*(2011), 69–84.

McGraw_Hill_construction. (2010). *The Business Value of BIM in Europe*. Columbus: The McGraw-Hill Companies.

Moore, ME. (2011). *Basics of Game Design*. Boca Raton: A K Peters/CRC Press Taylor & Francis Group.

NBIMS. (2007). *United States National Building Information Modeling Standard Version 1 - Part 1 - overview principles and methodologies*. Washington, DC: National Institute of Building Science.

Obituary. (2011). Alexander (Sandy) Shafto Douglas 1921–2010. *Comput J, 54*(2), 187–188.

Petri, I, Li, H, Rezgui, Y, Yang, C, Yuce, B, & Jayan, B. (2014). *A HPC based cloud model for real time energy optimization, Enterprise Information Systems*. UK: Taylor & Francis.

Revit 2012 API. (2011). *Developer's Guide*. United State: Autodesk, Inc.

Rüppel, U, & Schatz, K. (2011). Designing a BIM-based serious game for fire safety evacuation simulations. *Adv Eng Inform, 25*(2011), 600–611.

Shiratuddin, MF, & Thabet, W. (2011). Utilizing a 3D game engine to develop a virtual design review system. *Journal of Information Technology in Construction (ITcon), 16*(2011), 39–68.

Stuart, K (2011). Modern Warfare 3 smashes records: $775m in sales in five days. Accessed 2 December 2014. [http://www.theguardian.com/technology/2011/nov/18/modern-warfare-2-records-775m] website

Sun, L, Fukuda, T, & Resch, B. (2014). A synchronous distributed cloud-based virtual reality meeting system for architectural and urban design. *Frontiers of Architectural Research, 3*(4), 348–357.

Wang, B, Li, H, Rezgui, Y, Bradley, A, & Ong, HN. (2014). BIM based Virtual Environment for Fire Emergency Evacuation. *The Scientific World Journal, 2014*, 589016. doi:10.1155/2014/589016.

Yan, W, Culp, C, & Graf, R. (2011). Integrating BIM and gaming for real-time interactive architectural visualization. *Autom Constr, 20*(2011), 446–458.