CIB World Building Congress 2019
Hong Kong SAR, China
17 – 21 June 2019

Prototyping for Real-time Heavy Lift Simulation using Game Engine System

Zhen Lei
Assistant Professor, Department of Civil Engineering, University of New Brunswick, Head Hall, 17 Dineen Drive, Fredericton, NB, Canada E3B 5A3.
(email: zlei@ualberta.ca)
Ming-Fung Francis Siu
Assistant Professor, Department of Building and Real Estate, The Hong Kong Polytechnic University, 11 Yuk Choi Rd, Hung Hom, Kowloon, Hong Kong.
Tel: (+852) 2766-5820 (email: francis.siu@polyu.edu.hk)
Christoph Sydora
Research Assistant, Department of Civil and Environmental Engineering, Faculty of Engineering, University of Alberta, Edmonton, Alberta, Canada.
(email: csydora@ualberta.ca)
SangHyeok Han
Assistant Professor, Building, Civil, and Environmental Engineering, Concordia University, 1455 Boulevard de Maisonneuve O, Montréal, Quebec, Canada.
(email: sanghyeok.han@concordia.ca)
Ulrich Hermann
Construction Engineering Manager, PCL Industrial Management Inc., 5402 99 St NW, Edmonton, Edmonton, Alberta, Canada.
(email: RHHermann@pcl.com)

Abstract

The process of construction for industrial projects often relies on large mobile cranes to lift heavy modules. It is a process involving in-depth engineering analysis in which lift engineers devote considerable effort to ensuring the safety and ease of the lifting process by generating lift studies. Due to the complexity of the site conditions, the lifting process has inherent uncertainties, and judgments are often made intuitively. Over the past decade, visualization has been used to simulate crane lifts in order to add certainty and confidence to the engineering plan and to assist in training crews. This paper introduces a newly developed heavy lift simulator using game engine which allows the user to interactively analyze the lifting process. Compared with 4D visualization, this development adds the dimension of real-time interactive simulation, which can be further extended to virtual reality (VR) applications in future research. The developed prototype is tested and validated using the case study of a heavy industrial project with single-player mode.

Keywords: Mobile crane; heavy lift plan; game engine; virtual reality; visualization

1. Introduction and Literature Review

In Alberta, Canada, heavy industrial developments are an important factor in driving the local economy due to the rich oil sands reserves. However, the construction process has been challenged low productivity due to the harsh climate in Alberta, among other factors. These mega projects are often located in remote areas where labour is not readily available. To improve the efficiency of construction, the modular construction method has been adopted and has proven to be suitable for fast-paced heavy industrial projects. In this paradigm, the entire project is broken down into modules that are prefabricated off site and shipped to the site directly for installation, a process which minimizes the need for labour and heavy equipment utilization. Construction companies in heavy industrial sector have strived to develop day-to-day innovative engineering tools to achieve highly efficient planning and logistics for heavy lifting (Hermann *et al.*, 2010a; Lei *et al.*, 2013; Lei *et al.*, 2013; Taghaddos *et al.*, 2014). Part of this development has focused on how to make heavy lift planning more efficient, given the challenge of lengthy turnaround time for engineering design cycles. Automation has been achieved for drafting, engineering planning, and lift visualization (Lei *et al.*, 2016). This paper will introduce a recently developed simulator for heavy lifts using a Unity game engine system, which has been used for lift validation and safety training purposes. The benefits of this simulation are that it allows the users to: (i) interact with the 3D environment; (ii) identify potential lifting hazards and perform collision detection; and (iii) simulate the lifting process and find the "ideal" lifting solution; and (iv) provide trainings for rigging crews and crane operators.

Crane-related analysis/research is not novel in the construction engineering research field. Efforts have been made in recent decades in academia and industry to develop algorithms to improve crane lift design. Due to the development of computer technology, these algorithms are programmed into applications to achieve automation. In general, crane-related research can be categorized into the following categories: (i) crane type and location selection (Hermann *et al.*, 2010a; Huang, Wong and Tam, 2011; Safouhi *et al.*, 2011; Lien and Cheng, 2014); (ii) crane path planning and simulation (Reddy and Varghese, 2002; Ali, Babu and Varghese, 2005; Chang, Hung and Kang, 2012; Lin *et al.*, 2014); (iii) crane lift visualization (Hornaday *et al.*, 1993; Lin *et al.*, 2012; Zhang and Hammad, 2012; Juang, Hung and Kang, 2013); and (iv) crane lift engineering design and analysis (Wu *et al.*, 2011; Lin *et al.*, 2012; Hasan *et al.*, 2013). From the past research, a trend can be seen that crane lifts increasingly rely on computer-aided technology for planning and design: from its original 2D plan to 3D lift planning and 4D visualization. It is more common in practice that engineering algorithms are embedded into the computer system to achieve automated analysis. Engineers prefer a more direct representation rather than the static designs of the past. Lift planning is also further assisted with technologies such as lasers (Lee *et al.*, 2009; Hwang, 2012). However, in previous work, crane simulations have been pre-planned/programmed such that the users are not able to interact in real-time with visualizations. This limits the users from exploring the potential lifting solutions and understanding the challenges of the lifting process. The game engine provides a dynamic environment for crane simulation, meaning that the users can control the movement of the crane and walk through the lifting process. This not only provides the benefits of validating the engineering planning, but also enhances the user's understanding of the lift. Fig. 1 shows the transition from 2D lift study to real-time crane simulation over time, along with computer technology updates. The real-time simulation can be used as a training tool for the crane operator, rigging crews and site staff, while examples of such training in construction can be seen for both education and crane operator versions: improving safety and avoiding unnecessary jobsite hazards (Smith and Trenholme, 2009; Rezazadeh *et al.*, 2011; Goulding *et al.*, 2012; Guo *et al.*, 2012). The applications of game technology, Virtual Reality (VR), and Augmented Reality (AR) in construction are popular as they provide a direct representation of the project and add value to project coordination and planning. The basis for these applications is often CAD models or Building Information Modelling (BIM) (Wang *et al.*, 2013). An example of the use of CAD models to achieve VR is given by (Whyte *et al.*, 2000). Preconstruction planning can also benefit from VR technology (Waly and Thabet, 2003). In industry, there are simulation systems that use 3D visualization. Liebherr, for instance, has implemented a crane training system, called LiSIM, for the LB 28 rotary drilling rig and LR1300 crawler crane (www.lisim.liebherr.com). Another recent work done in this field is called the ITI VR

mobile crane simulator, a tool which integrates game engine, VR technology, and controlling hardware (https://www.iti.com/vr). Simulation using visualization is increasingly prevalent in construction management in general for the purpose of trainings on high-risk construction activities. CM Labs has developed a Vortex Studio Platform that can be used for building various simulators. The Vortex Studio Platform allows the users to model the component from scratch in order to fully create a construction scene (https://www.cm-labs.com/). However, this paper proposes a system that allows multi-player interactions during the simulation process, thereby enhancing the simulation process and achieving a greater degree of realism.

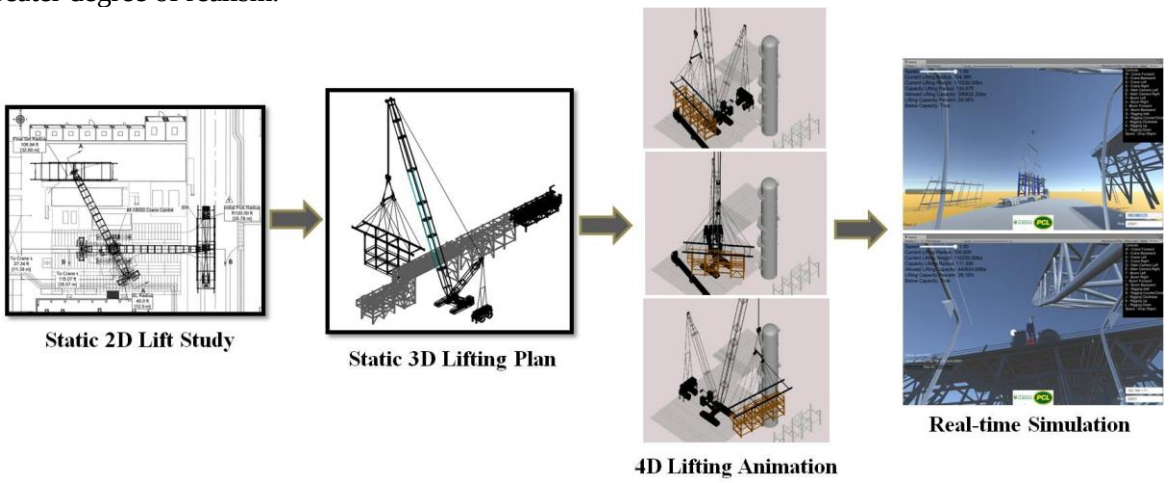

Figure 1: Progression of crane lift planning system development

As the base for game engine development, BIM has been widely used in construction, from planning and design to onsite construction management. Autodesk® tools (e.g., AutoCAD®, Revit®, 3ds Max®) have been adopted by builders and designers in industry. Efforts have been made to implement BIM in both onsite and offsite construction (e.g., workplace safety analysis, 4D visualization, etc.) (Akinci, Fischer and Kunz, 2002; Hu and Zhang, 2011; Nawari, 2012; Zhang *et al.*, 2013). The data-driven nature of BIM lays a foundation for game and simulation development, as can be observed in this research. We begin with AutoCAD static models from crane lift studies and export them as .fbx files, a file format compatible with Unity. These exported files serve as object components in Unity. C# scripts are then created to link the model to its physical movements. We also integrated crane lifting capacity charts given by the crane manufacturer with the dynamic game system; this allows the users to check the lifting capacity during the simulation. Ultimately, clash detection is used to ensure that the users understand the potential hazards during the crane lifting process. This research thus explores the opportunity to integrate BIM with game engine to achieve dynamic simulation of the lifting process. It is believed that a fine data model is necessary in this development, and in the future, we intend to explore the potential for VR application in the developed game environment. The ultimate goal of this research is to extend existing BIM technology to a more direct and interactive approach to project coordination and management. A real case study is provided to demonstrate how the proposed system is developed and implemented.

2. System Development

The development originates from a central SQL database where all the project and engineering data is stored. The idea is to provide a central data source for all the engineering planning to facilitate easy communication among various parties in the project. From there, applications have been developed specifically for onsite crane management: crane location selection and optimization (Hermann *et al.*, 2010b), lifting sequencing (Taghaddos *et al.*, 2014), 4D animations from macro and micro perspectives (Lei *et al.*, 2016), and automated client-oriented lift study CAD system. In the central database, three main types of data are stored: (i) the project data: lifting object specifications (e.g., dimensions, weight, etc.); (ii) crane data (e.g., crane dimensions, lifting capacity chart, etc.); and (iii) rigging data (e.g.,

rigging components such as shackles, slings, etc.). The pre-step before this study is the static lift study and 3D crane simulation, which create the crane/module/environment models. However, these models are not capable of linking to one another to function in a 3D space, which is achieved in the Unity environment.

2.1 Game Engine Development

The game engine used in this study is Unity®. Unity is a software primarily used in the video game industry to develop computer games. Other applications of Unity include mobile apps and website and console games. The software has a built-in 3D Physics engine that can help simulate the way objects would interact in the real world. With the use of scripting, customizable behaviour, and object control can be achieved to suit the need of the application. There are other platforms available for game development, (e.g., Unreal®), most of which provide scripts or application program interface (API). We select Unity® as the development tool due to its ease to use and large supporting community. As shown in Fig. 2, in Unity, the foundation for any development is the scene, which contains other game objects. Inside a scene, the environments, obstacles, and objects can be placed by the users. The scene serves as a "global environment" that holds all game items inside. The game objects are literally the objects in the game environment, and can be 3D/2D physical objects or dummy objects. In this research, 3D game objects are used in the development. However, these game objects cannot function if properties are not assigned to them; once assigned, they become characters in the game environment (e.g., tree, light, crane boom). The game objects can interact with one another, or they can themselves contain multiple game objects to make themselves nested objects (as described by the 1:N relationship). The "properties" that help the game objects function in 3D space are called components. For example, the transform component defines the position, rotation, and scale of a game object in the 3D scene. Typical components used in our study include: translation, rigid body, script, collider, and joint, as will be further elaborated on later in this paper.

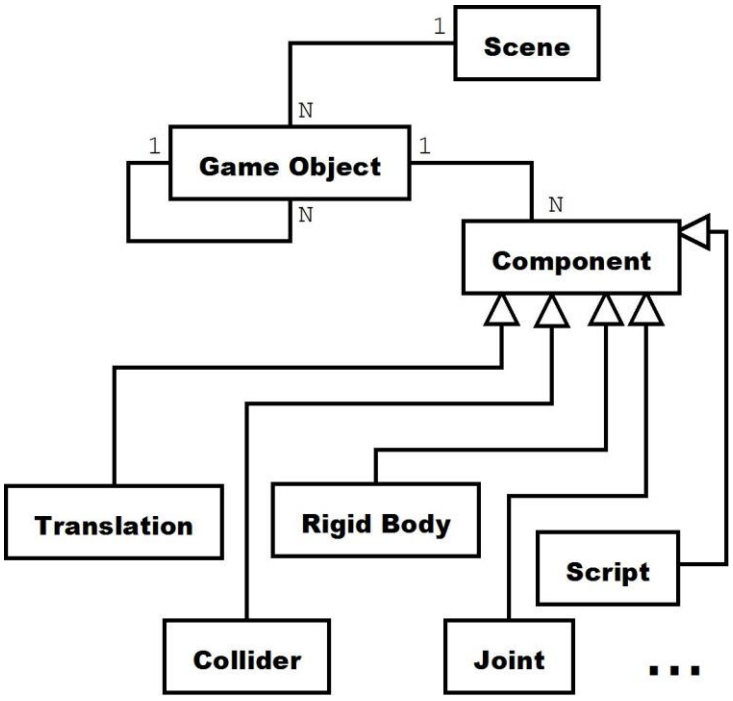

Figure 2: Unity game engine development structure

In Unity, game objects are the base class for all objects in the scene. A game object can represent any real-life 3D entity (e.g., box, tube). In this research, a crane boom, mast, and crawler are each separate game objects. They can be grouped together into one parent game object. However, the crane objects are not able to move, collide, or interact in any way without containing any components. For a typical

crawler crane, the game objects can be defined based on its degrees-of-freedom (DOFs), namely, the number of independent parameters to define its configuration in a 3D environment. Fig. 3 shows a typical crawler crane's DOFs and corresponding game objects used in game development. The defined DOFs also set the movement types of the crane in the game environment. For other crane types, DOFs can be defined differently in implementation.

$$DOFs = \{\alpha, \beta, \gamma, x, z\} \quad (1)$$

where α = rotation of the rigging and lifting module; z = booming up and down; β = rotation of the crane superstructure; x = crawler walking forward and backward; and γ = hoisting up and down.

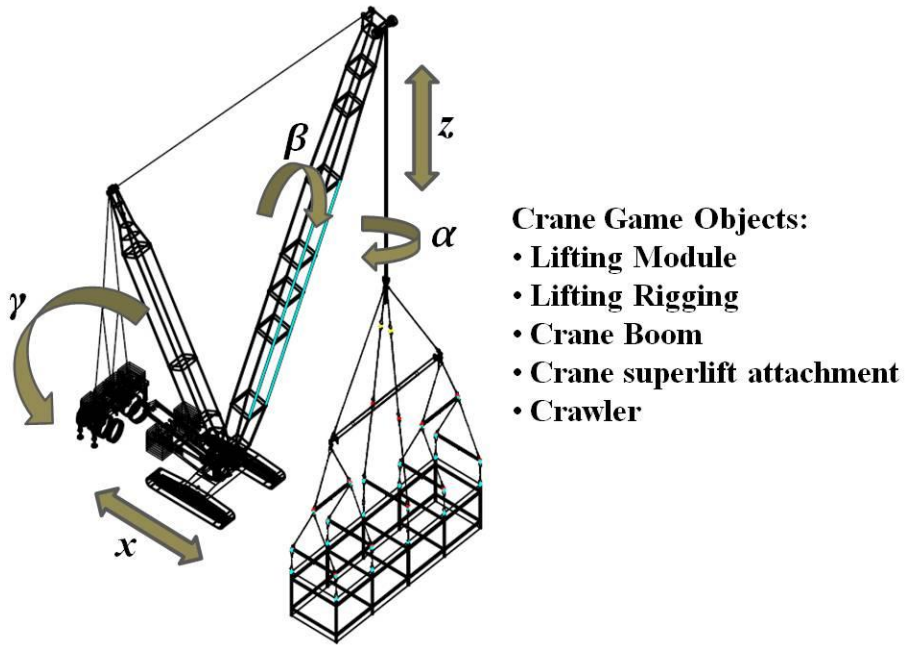

Figure 3: Degrees-of-Freedom (DOFs) of a typical mobile crane

Components are the defining attributes of each game object that determine the role of the game object in the scene. All game objects inherit from a standard object class that contains only the transform component: the object's location, rotation, and scale in a scene. Also, in Unity, any game object can be nested within a parent object, and by doing so, its transform components become relative to its immediate parent game object. Therefore, the global location of an individual game object would be the local position of the object multiplied by the translation matrix of the parent game object. To give a generic example, if an object is the child of an object whose location is rotated about the y-axis by θ and translated by a vector of $[t_x, t_y, t_z]$, then the location of the child object globally can be calculated using the following homogeneous transform:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & t_x \\ 0 & 1 & 0 & t_y \\ -\sin\theta & 0 & \cos\theta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2)$$

In the Unity system, the local and global locations of the object can be found using the Transform.localPosition and Transform.position properties of the game object class, respectively. Many different types of components can be added to a single game object to give different functionalities within the scene. For the crane simulation development, some of the key components required are outlined as follows: (i) The mesh component is the "skin" of the object and gives the object its form,

(although without a collider component other objects would simply pass through the object). (ii) The collider component is used to detect collisions between objects in 3D space. Collider components can take on primitive shapes (such as capsule, sphere, or box colliders) or can use the mesh of an object to take the exact shape of the object (mesh collider). Unity has a built-in Ray Casting mechanism to check collisions, such that reducing the face count of the objects will result in a faster processing time. (iii) The rigidbody component makes use of physics simulations such as gravity. This also facilitates the functioning of the collider. In order for an entire object to react to forces in the scene, each child game object is given a rigidbody component which takes in the mass and other kinematic properties of the object. (iv) A joint component is a direct connection between two game objects, specifically the two rigidbodies of each game object. These are added to an object to adjust it so that the combined objects' DOFs accurately match those of the desired structure. By modifying the orientation of the pivot of a joint component, the axis around which the two objects can rotate with respect to each other can be altered.

In addition to the above-mentioned components, scripts components serve as means of controlling game objects over short iterations in a way that cannot be achieved using the other components alone (e.g., movement, rotation). All scripts inherit from a base class MonoBehaviour. MonoBehaviour contains predefined methods that are executed automatically depending on the function of the method. There are four main methods of MonoBehaviour as listed in Table 1. When certain events are triggered, an object's behaviour and properties can be programmatically altered as needed in the scene. For example, when two objects' colliders come into contact, an event would be triggered; forces or components can be applied to the objects in order to mimic any type of effect, such as an explosion or a fusion of the objects, using joints. Using events that can be set to trigger over certain time intervals allows for the addition of controls to the game object by checking keys that are pressed. For example, when a keyboard key is pressed, the object will respond to the corresponding action such as by booming up or down. Scripting is also essential for network management and for controlling the network flow and synchronization in the scene.

Table 1. Sub-classes of Unity Script

Method Name	***Function***
Start()	*Called once at the start of the scene to initialize the game environment*
FixedUpdate()/Update()	*Called over a fixed time interval or over each frame to update the configurations of the game objects in the game environment*
OnCollisionEnter()/OnCollisionStay()/ OnCollisionExit()	*Called when two colliders come into contact, are currently in contact, or have separated from each other*
OnDestroy()	*Called once when the scene is terminated*

3. Implementation and Case Studies

The crane lift study is broken down into separate objects that can be imported into Unity as .fbx files. The objects include: crane mats, crane boom, hoist cable linking rigging to boom, crane superlift attachment, crane crawler, mast and derrick, surrounding environment and obstacles, lifting module, and delivery truck. After importing, components are added and the game objects are arranged and aligned as needed (a hierarchy is created to link the game objects together). Fig. 4 shows the Unity interface with the imported model. The interface contains the edited window and the game play testing window (i.e., the field of vision of the camera). The hierarchy defines the game object structure, and, in this study, a main game object "crane" is created to include all crane objects: crane crawler track, derrick mast, superlift counterweight, rigging, and cables. The rationale here is that, rather than using separate scripts to control individual crane objects, a master script can be used to control all sub-objects through

the main "crane" object. The hierarchy also defines the DOFs of the crawler crane: the crawler crane track is at a higher hierarchical level, as the movement of the crawler track dictates the other crane superstructures (e.g., boom, mast) versus the movement of any superstructures, for example boom-up, does not trigger the movement of the crawler track. Moreover, all crane objects interact with one another through the Unity component Joint.

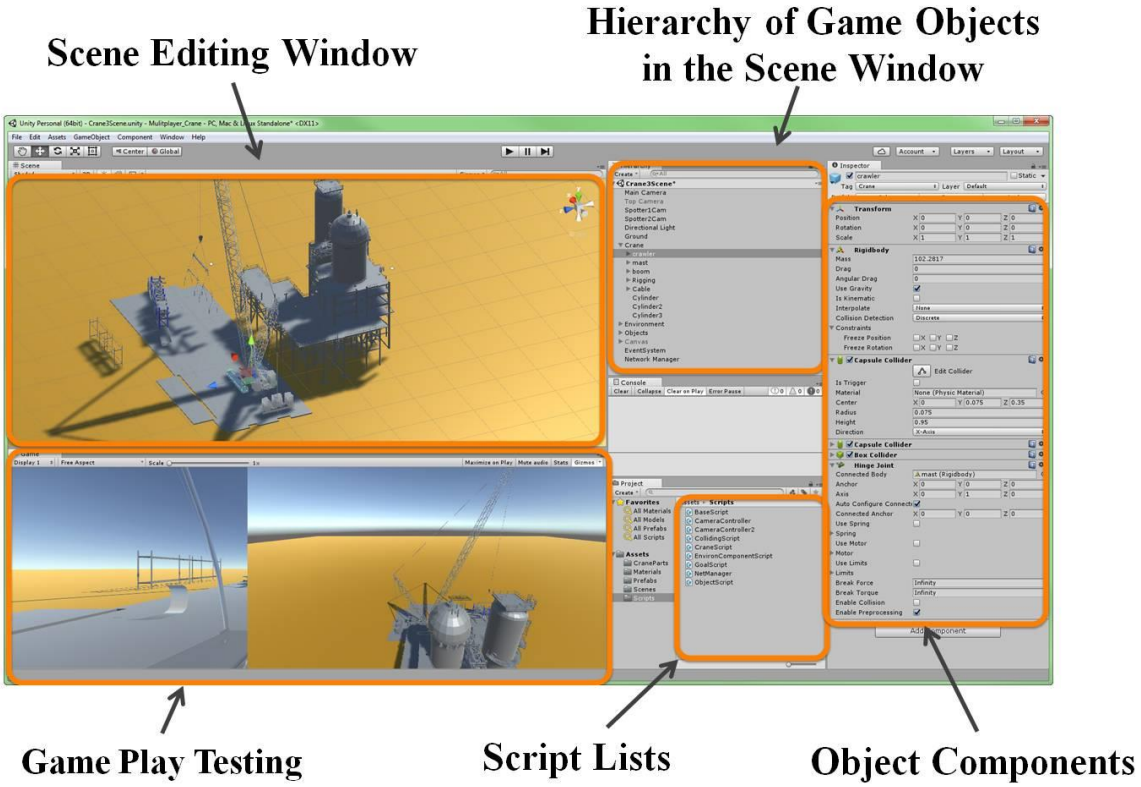

Figure 4: Unity Interface with Imported 3D Model

The *Scripts* are used to control crane components through the *Joints*. In this research, *Scripts* can be categorized as follows: (i) crane movement script, which controls the crane movements; (ii) camera control script, which allows the user to move the camera and view the project from difference perspectives; and (iii) collision/complete event trigger script, which detects any collisions in the lifting process and completes the lifting once the object reaches its set location. Meanwhile, there are two types of crane movements—translation and rotation—each of which can be controlled by the *GameObject.Transform* function, a built-in function of Unity API. The *GameObject.Transform* allows the users to access the object's 3D position, rotation angle, and scale. For example, for the crane crawler, forward movement can be controlled through the crawler's *rigidbody* (`private Rigidbody crawlerRB`), and, when the `KeyCode` CrawlerForward is pressed, the updated crawler position can be redefined by a `Vector3` in FixedUpdate():

```
crawlerRB.transform.position = new Vector3
(crawlerRB.transform.position.x + crawlerObj.transform.forward.x,
crawlerRB.transform.position.y + crawlerObj.transform.forward.y,
crawlerRB.transform.position.z + crawlerObj.transform.forward.z);
```

In addition, during the lifting process, the collision between the crane and the surrounding environment is continuously checked through the *Collider* object. *Collider* objects can be defined in the Unity environment and checked against one another using the `void OnCollisionStay(Collision collision)` function through the game objects' nametags.

3.1 Crane Lifting Simulation in Unity Environment

In the presented case, a real critical lift is used for testing the developed game engine simulation. As shown in Fig. 5, a LR-1600 crane with buggy superlift attachment is used to perform this lift. The total weight is 114,017 lb (51.7 Te) (including module load weight, load block weight, rigging weight, and auxiliary ball and runner). At maximum radius 115 ft (35.1 m), 88.4% crane capacity is used, given that the manufacturer's chart capacity is 128,970 lb (58.5 Te). As shown in Fig. 5, the module is picked up from the delivery truck and swung clock-wise to the set position. The static lift study in Fig. 5 and 6 does not give a clear understanding of potential collisions between the lifted object and existing structures (shaded objects in the figure) at the final set position. In such a case the crane operator and other onsite personnel are thus required to "imagine" the trajectory for the lifted module. An elevation view is provided to show the potential collision at the final position between the lifted module and the surround structures (Fig. 6). However, the elevation view also shows merely a static scene with the lifted module at its final set location for the purpose of clearance checking. In Unity, the lifting scenario is created as shown in Fig. 7. Through the keyboard control, the user is able to maneuver the lifted object to its set location, during which the lifting capacity is checked at each time frame, and collision detection is performed (as in Fig. 8, where the lifting radius is 130.65 ft with a lifting capacity of 30.77%). Once the lifted object reaches its set location (Fig. 9), the Script is triggered and the user is signaled that the lift is "Complete".

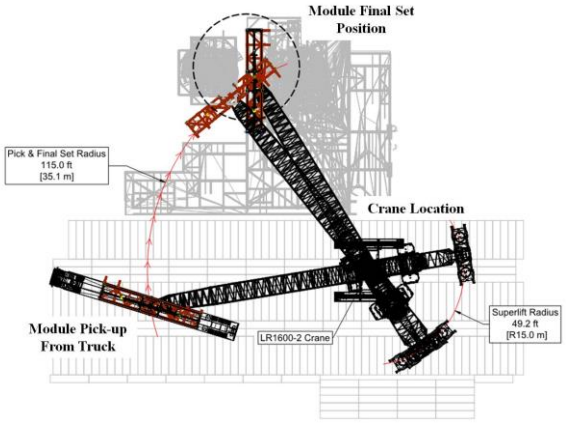

Figure 5: Plan view of a critical heavy lift study

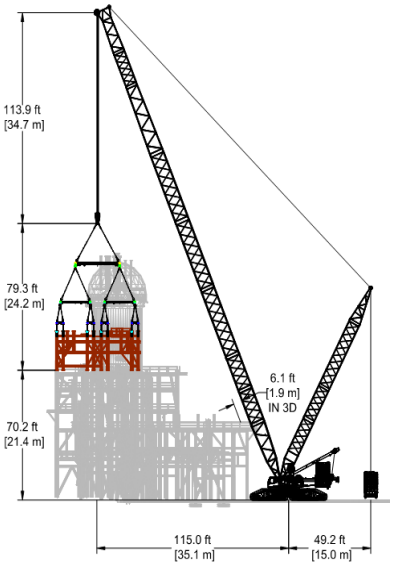

Figure 6: Elevation view of a critical heavy lift study

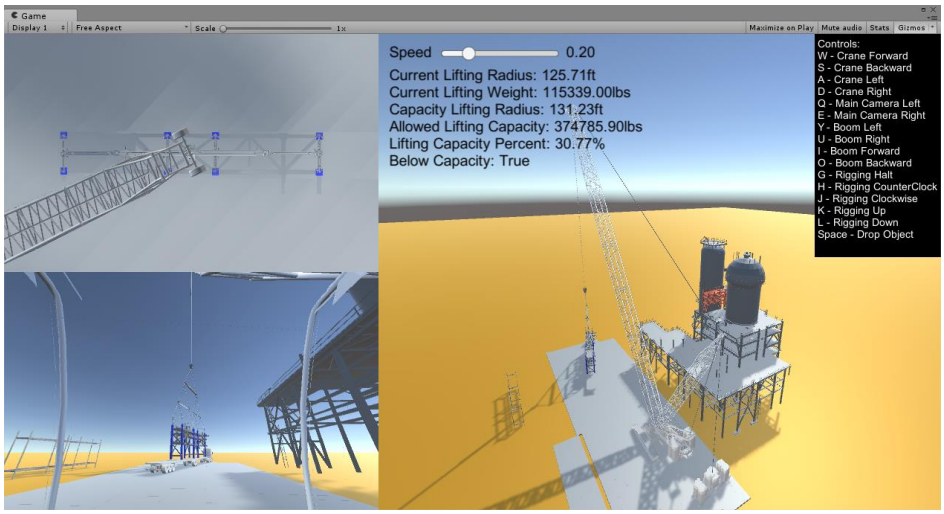

Figure 7: Lifting scenario in Unity environment

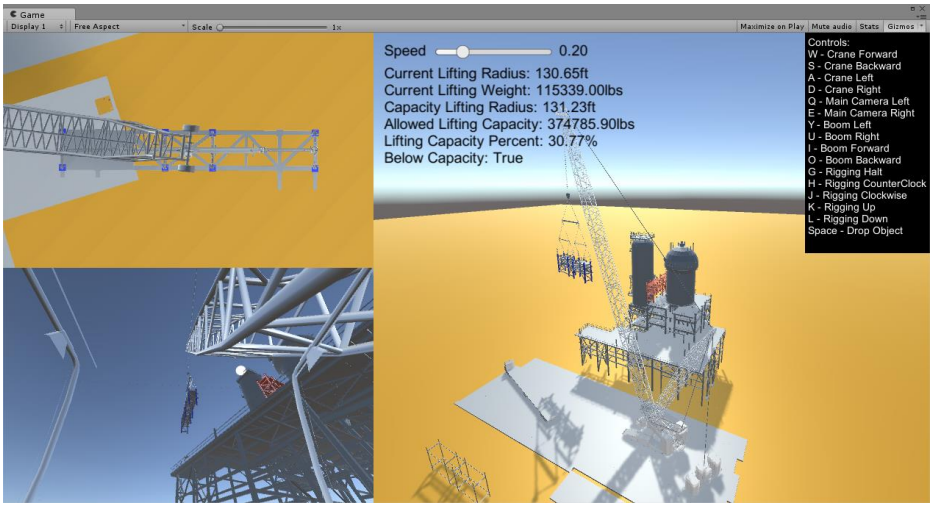

Figure 8: Crane lifting module simulation

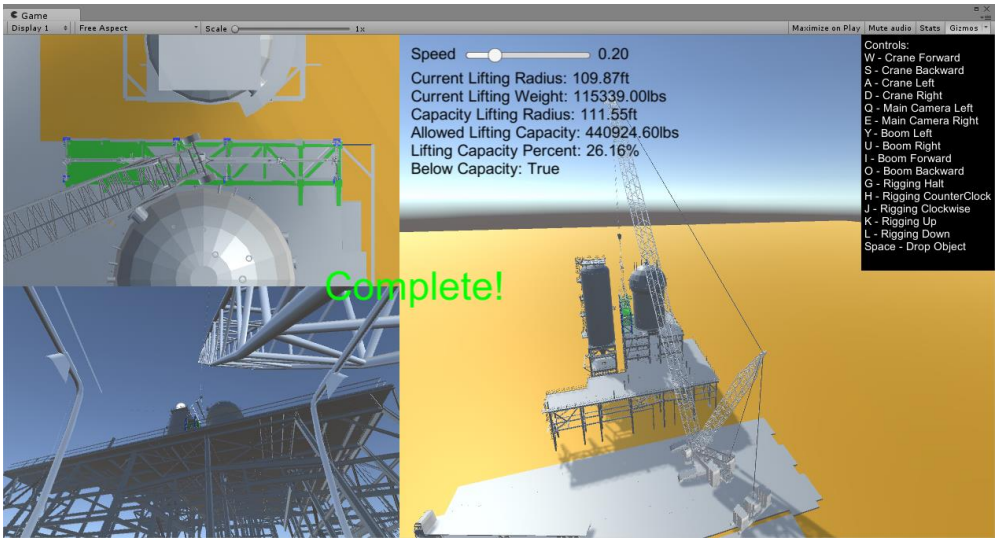

Figure 9: Crane lifting complete at the module set location

4. Conclusions and Future Works

In this paper, a prototype crane simulator developed using Unity Engine was introduced. The system considers the DOFs of the crane and rigging and embeds the lifting capacity. Given that the simulator allows the user to control the crane movement, a physics mechanism was implemented to achieve the interaction among the rigid bodies of these crane components. Colliders are added to help detect the collision and determine the completion status of the lift. In the methodology and implementation, algorithms and related API were introduced that other researchers/practitioners can refer to in future work or implementation. One case study was provided to validate the proposed method and the efficiency of the simulation. The single-player crane operation is presented with the lifting capacity calculation and lifting radius during the lift. From the single-player mode, it can be observed that there are a few blind spots where the crane operator will have difficulty operating the lift without the assistance of signal persons. Although the system is limited in that each lifting scenario requires manual setup and adjustment of the model, this can be automated through API codes in the future to reduce the human effort in the process of creating the simulation. In addition, this work can be further extended to VR to achieve first-person simulation. Existing equipment such as Oculus and Vive can be used for this purpose. Also, this research can be applied as a simulation tool to assist construction practitioners in training and in improving their onsite safety performance, and same concept can be used in workplace safety training. Furthermore, existing path planning/finding algorithms for crane lifting can be easily validated and visualized using this research.

5. Acknowledgements

The authors would like to acknowledge all the participants in this research, particularly our industry collaborator, PCL Industrial Management Inc., and the Natural Sciences and Engineering Research Council of Canada (grant no. CRDPJ 491811-15) for their funding support. We also appreciate the support received from the Hole School of Construction Engineering and from the Nasseri School of Building Science and Engineering at the University of Alberta.

References

Akinci, B., Fischer, M. and Kunz, J. (2002) 'Automated Generation of Work Spaces Required by Construction Activities', *Journal of Construction Engineering and Management*, 128(4), pp. 306–315. doi: 10.1061/(ASCE)0733-9364(2002)128:4(306).

Ali, M., Babu, N. and Varghese, K. (2005) 'Collision free path planning of cooperative crane manipulators using genetic algorithm', *Journal of Computing in Civil Engineering*, 19(2), pp. 182–193. Available at: http://ascelibrary.org/doi/abs/10.1061/(ASCE)0887-3801(2005)19:2(182) (Accessed: 28 May 2014).

Chang, Y.-C., Hung, W.-H. and Kang, S.-C. (2012) 'A fast path planning method for single and dual crane erections', *Automation in Construction*. Elsevier B.V., 22, pp. 468–480. doi: 10.1016/j.autcon.2011.11.006.

Goulding, J. *et al.* (2012) 'Construction industry offsite production: A virtual reality interactive training environment prototype', *Advanced Engineering Informatics*. Elsevier Ltd, 26(1), pp. 103–116. doi: 10.1016/j.aei.2011.09.004.

Guo, H. *et al.* (2012) 'Using game technologies to improve the safety of construction plant operations', *Accident Analysis and Prevention*. Elsevier Ltd, 48, pp. 204–213. doi: 10.1016/j.aap.2011.06.002.

Hasan, S. *et al.* (2013) 'Productivity and CO2 emission analysis for tower crane utilization on high-rise building projects', *Automation in Construction*, 31, pp. 255–264. doi: 10.1016/j.autcon.2012.11.044.

Hermann, U. *et al.* (2010a) 'An Integrated System to Select, Position, and Simulate Mobile Cranes for Complex Industrial Projects', in *Construction Research Congress 2010*. Reston, VA: American Society of Civil Engineers, pp. 267–276. doi: 10.1061/41109(373)27.

Hermann, U. *et al.* (2010b) 'An Integrated System to Select, Position, and Simulate Mobile Cranes for Complex Industrial Projects', *Construction Research Congress 2010*, pp. 267–276. doi: 10.1061/41109(373)27.

Hornaday, W. C. *et al.* (1993) 'Computer-aided planning for heavy lifts', *Journal of Construction Engineering and Management*, 119(3), pp. 498–515.

Hu, Z. and Zhang, J. (2011) 'BIM- and 4D-based integrated solution of analysis and management for conflicts and structural safety problems during construction: 2. Development and site trials', *Automation in Construction*, 20(2), pp. 155–166. doi: 10.1016/j.autcon.2010.09.013.

Huang, C., Wong, C. K. and Tam, C. M. (2011) 'Optimization of tower crane and material supply locations in a high-rise building site by mixed-integer linear programming', *Automation in Construction*. Elsevier B.V., 20(5), pp. 571–580. doi: 10.1016/j.autcon.2010.11.023.

Hwang, S. (2012) 'Ultra-wide band technology experiments for real-time prevention of tower crane collisions', *Automation in Construction*. Elsevier B.V., 22, pp. 545–553. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0926580511002226 (Accessed: 28 May 2014).

Juang, J. R., Hung, W. H. and Kang, S. C. (2013) 'SimCrane 3D+: A crane simulator with kinesthetic and stereoscopic vision', *Advanced Engineering Informatics*. Elsevier Ltd, 27(4), pp. 506–518. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1474034613000529 (Accessed: 23 May 2014).

Lee, G. *et al.* (2009) 'A laser-technology-based lifting-path tracking system for a robotic tower crane', *Automation in Construction*. Elsevier B.V., 18(7), pp. 865–874. doi: 10.1016/j.autcon.2009.03.011.

Lei, Z., Taghaddos, H., Hermann, U., *et al.* (2013) 'A methodology for mobile crane lift path checking in heavy industrial projects', *Automation in Construction*. Elsevier B.V., 31, pp. 41–53. doi: 10.1016/j.autcon.2012.11.042.

Lei, Z., Taghaddos, H., Olearczyk, J., *et al.* (2013) 'Automated Method for Checking Crane Paths for Heavy Lifts in Industrial Projects', *Journal of Construction Engineering and Management*, pp. 04013011-1–9. doi: 10.1061/(ASCE)CO.1943-7862.0000740.

Lei, Z. *et al.* (2016) 'From AutoCAD to 3ds Max : An automated approach for animating heavy lifting studies', *Canadian Journal of Civil Engineering*, 1(37), p. 5404. doi: 10.1139/cjce-2014-0313.

Lien, L.-C. and Cheng, M.-Y. (2014) 'Particle bee algorithm for tower crane layout with material quantity supply and demand optimization', *Automation in Construction*. Elsevier B.V., 45, pp. 25–32. doi: 10.1016/j.autcon.2014.05.002.

Lin, Y. *et al.* (2012) 'Statics-based simulation approach for two-crane lift', *Journal of Construction Engineering and Management*, 138(10), pp. 1139–1149. Available at: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CO.1943-7862.0000526 (Accessed: 28 May 2014).

Lin, Y. *et al.* (2014) 'Lift path planning for a nonholonomic crawler crane', *Automation in Construction*, 44, pp. 12–24. doi: 10.1016/j.autcon.2014.03.007.

Nawari, N. O. (2012) 'BIM Standard in Off-Site Construction', *Journal of Architectural Engineering*, 18(2), pp. 107–113. doi: 10.1061/(ASCE)AE.1943-5568.0000056.

Reddy, H. R. and Varghese, K. (2002) 'Automated Path Planning for Mobile Crane Lifts', *Computer-Aided Civil and Infrastructure Engineering*, 17(6), pp. 439–448. doi: 10.1111/0885-9507.00005.

Rezazadeh, I. M. *et al.* (2011) 'Using affective human-machine interface to increase the operation performance in virtual construction crane training system: A novel approach', *Automation in Construction*. Elsevier B.V., 20(3), pp. 289–298. doi: 10.1016/j.autcon.2010.10.005.

Safouhi, H. *et al.* (2011) 'An algorithm for the calculation of feasible mobile crane position areas', *Automation in Construction*. Elsevier B.V., 20(4), pp. 360–367. doi: 10.1016/j.autcon.2010.11.006.

Smith, S. P. and Trenholme, D. (2009) 'Rapid prototyping a virtual fire drill environment using computer game technology', *Fire Safety Journal*, 44(4), pp. 559–569. doi: 10.1016/j.firesaf.2008.11.004.

Taghaddos, H. *et al.* (2014) 'Simulation-Based Multiagent Approach for Scheduling Modular Construction', *Journal of Computing in Civil Engineering*, 28(2), pp. 263–274. doi: 10.1061/(ASCE)CP.1943-5487.0000262.

Waly, A. F. and Thabet, W. Y. (2003) 'A Virtual Construction Environment for preconstruction planning', *Automation in Construction*, 12(2), pp. 139–154. doi: 10.1016/S0926-5805(02)00047-X.

Wang, X. *et al.* (2013) 'A conceptual framework for integrating building information modeling with augmented reality', *Automation in Construction*. Elsevier B.V., 34, pp. 37–44. doi: 10.1016/j.autcon.2012.10.012.

Whyte, J. *et al.* (2000) 'From CAD to virtual reality: Modelling approaches, data exchange and interactive 3D building design tools', *Automation in construction*, 10(1), pp. 43–45. doi: 10.1016/S0926-5805(99)00012-6.

Wu, D. *et al.* (2011) 'Algorithm of crane selection for heavy lifts', *Journal of Computing in Civil Engineering*, 25(1), pp. 57–65. Available at: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000065 (Accessed: 28 May 2014).

Zhang, C. and Hammad, A. (2012) 'Improving lifting motion planning and re-planning of cranes with consideration for safety and efficiency', *Advanced Engineering Informatics*. Elsevier Ltd, 26(2), pp. 396–410. doi: 10.1016/j.aei.2012.01.003.

Zhang, S. *et al.* (2013) 'Building Information Modeling (BIM) and Safety: Automatic Safety Checking of Construction Models and Schedules', *Automation in Construction*. Elsevier B.V., 29, pp. 183–195. doi: 10.1016/j.autcon.2012.05.006.