Provisional chapter

Virtual Simulation Platform for Training Semi-Autonomous Robotic Vehicles' Operators

Cheng Siong Chin, Xionghu Zhong, Rongxin Cui, Chenguang Yang and Mohan Venkateshkumar

Additional information is available at the end of the chapter

Abstract

This chapter covers the development of a virtual simulation platform for training a semi-autonomous robotic vehicle (SARV) operator via an open-source game engine called Unity3D. The SARV such as remotely operated vehicles (ROVs) is becoming increasingly popular in the maritime industry for risky jobs in inhospitable environments. The primary element in carrying out underwater missions in a hostile environment lies within the skills and experience of an ROV pilot. Training for ROV pilots is essential to prevent damage to expensive field equipment during the real operations. The proposed simulator differs from the existing simulators in the market is the use of modern game engine software to develop a "serious game" for ROV pilot trainee at much lower cost and shorter time-to-market. The results revealed that proposed virtual simulator can develop a high-fidelity virtual reality training for the underwater operation guided by classification society.

Keywords: autonomous robotic vehicle, Unity3D, remotely operated vehicle, simulation

1. Introduction

In recent years, the advancement of technology has dramatically improved the functions of remotely operate vehicle (ROV) [1–3] and autonomous underwater vehicle (AUV) [4] to handle the growing spectrum of underwater tasks [5]. Artificial Intelligence is becoming an increasingly common sight in automating machines to carry functions without the need for an actual operator. It will continue to take on a more observatory role [6–8]. Nowadays, ROVs and AUVs are commonly used in the maritime industry to carry out underwater tasks. These machines possess the capabilities to carry heavier loads to stay in deeper underwater for a longer

duration than the human divers. These machines are operated from shore by a pilot making them very profitable and safer in the maritime industry. The challenge facing these ROV pilots is the ability to run the ROV with minimal information from the ROV feedback systems. As a result, the ROV pilot needs to be sufficiently skilled in maneuvering the vehicle in underwater.

With modern day technology, training simulators are developed to better equip ROV pilots with the necessary skills. Currently, the majority of ROV simulators available in the market are owned and distributed by companies that build ROVs. It is much cheaper to hone a pilot's competencies on a simulator than on the actual ROV as it creates room for the pilot to improve. It would better equip the pilots to deal with different underwater scenarios. Many ROV simulators are equipped with multiple scenes and a wide array of simulated sensors and equipment on the ROV.

This paper aims to develop a virtual simulation where ROV pilots can gain experience via a virtual environment using an open-source game development software to produce highly graphical visuals simulations for training purpose. The Unity3D game engine [8–10] was identified as a suitable development platform for the project due to its high graphics capabilities, built-in physics engine, well-documented manual, large online community and relatively mild learning curve in comparison with other game engines like the Unreal Engine [11] and CryEngine [12]. The details of designing a low-cost pilot simulator using a game engine are unique to this chapter.

The chapter is organized as follows. Section 2 presents a brief methodology for the training simulator. Section 3 discusses the virtual simulation development and followed by a conclusion.

2. Comparison of game engines

Game engines have the graphics and physics engines to build better and more realistic simulation. In this section, three commonly used open-source game engines are compared. For example, the commonly used software for game development is namely: Unity3D, Unreal 4 Engine, and CryEngine.

There exists few open source game engine software that provides excellent features and developing tools. Some of the common characters of a game engine are rendering, physics (2D and 3D rigid body), scripting, audio and animation. Depending on the requirements, these game engine software use traditional programming method that requires basic coding to high-level sandbox engine that provides "drag and drops" interface. The main objective is to simulate an ROV operation using high-level sandbox game engine for the ease of usability. The options are more toward the sandbox engine and the more common software for game development such as Unity3D, Unreal Engine and CryEngine. Although the software provides developers with a "drag and drop" interface, the features of each software somehow differ from one another. A brief comparison of various game engines is given below.

Unity3D was first released in 2005. It uses mostly JavaScript or C# or managed code tool chain that makes it simpler to support and develop new workflows and tools. It has large supporting communities that include the asset store for downloading different game characters, particle

and sound effects. Due to its popularity, there exists a good educational material and large active user. However, the free version of Unity does not have Profiler that allows the programmer to optimize the game and check the time spent on rendering, and animation during the game. Unity3D supports around 21 platforms (PC, Web, Console, Mobile, etc.) as compared to Unreal Engine 4 supporting only around six platforms. Additionally, the 3D models in Unity3D can merely import as game assets into the software thus improving the efficiency of development.

Unreal Engine was first released in 1998. It provides developers with powerful tools such as access to full source code, simulates and immerse view, persona animation, and cascade visual effects. It is used in a custom workstation with better and optimized performance that implies higher cost and complexity. Unreal has much more extensive download than Unity3D as it requires visual studio for its programming environment and accepts only C++ development language. Unreal engine can produce high-quality graphics with advanced dynamic lightings making it a plus point for the game engine. However, the script used in Unreal Engine 4 can only be written in C++, which can be a drawback for beginners. Similar to Unity3D, Unreal Engine 4 has an asset store to download different game assets. However, the user community is not as large as Unity3D.

CryEngine started in 2002. It is another modern game engine that provides superb features that will create great realistic gameplay. With its pixel accurate displacement mapping, it allows developers to craft and modify a game as precise as possible. Its excellent graphics capabilities exceed those in Unity3D and Unreal Engine. However, a drawback from this game engine is that it requires a slightly higher learning curve before one can use the game engine efficiently and it may be harder for those with no game development background.

It is notable that these three game engines provide great features for the most development process. It can be quite subjective in the selection decision. Depending on the development objectives and requirements, one may pick Unity3D for its capabilities in developing 2D and 3D games, Unreal Engine for its powerful tools or CryEngine for its extreme graphics capabilities. Fortunately, these game engines are freely available for education and research except for Unity3D which requires Pro version for advanced features. On the other hand, the Unreal Engine and CryEngine require a slightly higher learning curve and posing difficulty for most beginners. Based on the following guidelines below, the free version of Unity3D that contains most of the functions will be used (at least for the beginning phase of the project) in developing a simulator for ROV pipeline tracking as it is easy to use, free for research purpose and the presence of wide user community. It may not be the best choice for every programmer, but during the development of the virtual reality simulator, there are no major problems encountered, and hence the choice was good enough.

- Able to communicate with external hardware
- Ease to program and use graphical user interface for controlling interaction and animating objects
- Able to process multimedia sensory data
- Free to use for education and research
- Able to hold multiple operating systems
- Able to support development with the strong developer community

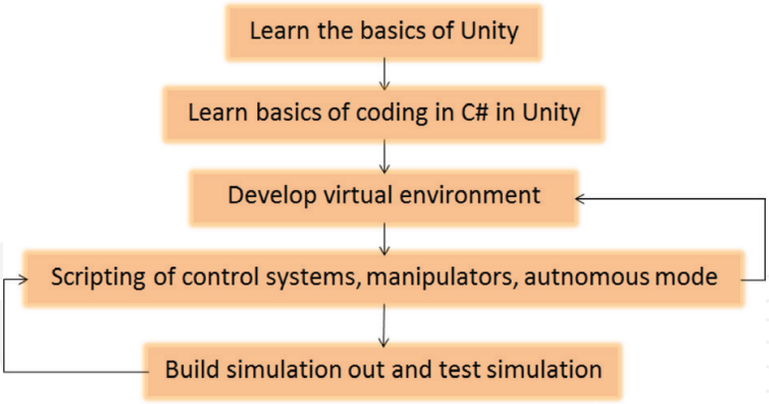

Figure 1. Steps to develop ROV pilot simulator.

The basic functions of the Unity3D Interface such as creating GameObjects, basics scripts and GameObjects for manipulation will be presented. The virtual environment and writing the scripts for the ROV's control system and manipulators will then follow. Assistance can be sought through the Unity3D manual found online or via Unity3D's online community and forums. As shown in **Figure 1**, the basic steps to develop the simulator can be seen. The simulator designed must be able to facilitate training of an ROV pilot. The environment of a typical pipeline inspection will be mimicked. After the leak is detected, the ROV will flash a bright red light indicating danger. The pilot will take control of the ROV and press the shutdown button of the BlowOut Preventer (BOP) to stop the leak and the flashing red light. After that, the pilot can continue the control of the ROV to carry out the autonomous inspection tasks. All controller inputs by the pilot are controlled via the joystick controller.

3. Virtual simulation development

Unity3D is a user-friendly off-the-shelf game development engine. The engine supports high visual graphics and physics to produce realistic 2D and 3D worlds, with readily developed assets available in the Unity3D Asset Store for users to download and import into their projects. A brief overall view of the software can be seen below.

3.1. Software interface

The Unity3D interface consists of several tabs and a toolbar (see **Figure 2**) are used to create all subsystems in the virtual simulation. The games are developed in the scene tab where GameObjects are added. These GameObjects are edited and programmed by combining various components such as textures, mesh, materials and scripts to make the GameObject behaves as required by the developer. As shown in **Figure 3**, the various components are placed into GameObjects under the virtual environment to produce the desired outcome.

3.2. Scene and model development

The scene developed in Unity3D is modeled after a subsea production system placed on the seabed 900 m below the surface. It consists of manifolds, pipelines, BOPs (BlowOut

Figure 2. Unity3D interface.

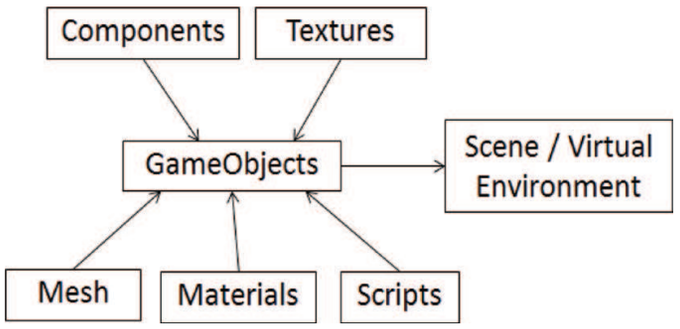

Figure 3. GameObject with various components.

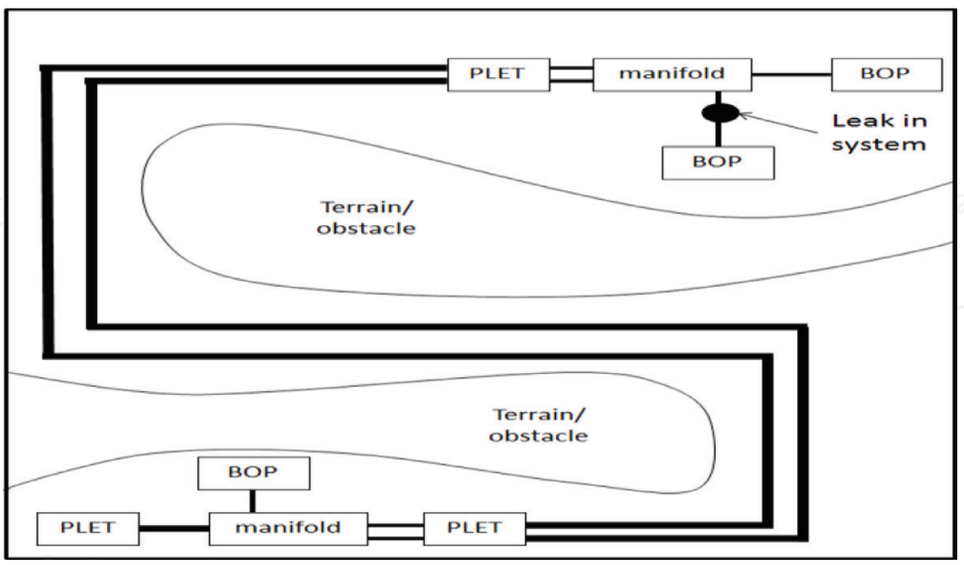

Figure 4. Map of simulated environment.

Preventer) and PLETs (Pipeline End Termination). A map of the simulated environment is shown in **Figure 4**. The scene was developed using models found online and crafted within Unity3D.

The ROV used in the simulation was modeled using an actual ROV used in the industry to fulfill requirement stated by the classification society DNV. It indicates that ROV models available in the simulation should be similar to the ROV in real life. In this proposed simulator, the TRV-M is a light work class ROV manufactured by Submersible Systems. The TRV-M in **Figure 5** can dive up to 1000 m deep that possesses the capacity to carry a payload up to 27 kg in any environments.

3.3. Autonomous operation

Automation is an area currently being developed in the maritime industry. The autonomous-mode component was added to the ROV in the simulator. In the simulation, the ROV begins in its autonomous inspection of the pipeline. It was achieved using a Unity3D function known as pathfinding where the object automatically computes the shortest path and moves itself to the desired point while avoiding the obstacles in the virtual environment. Using the pathfinding algorithm, multiple waypoints were added along the path of the pipeline to guide the ROV during the inspection.

3.4. Controller

To create a more realistic simulator, the control system for the ROV in the simulator should be similar to the control system of the ROV. A joystick controller or keyboard is one of the common controllers used in the maritime industry. The joystick controller will require multiple turning axes to accommodate all six degrees of motion similar to the movement of a vessel. The simulated ROV is capable of moving in six degrees of motion that is heave, sway, surge, yaw, pitch, and roll in **Figure 6**. As shown in **Figure 7**, the Logitech Extreme 3D Pro is used as the joystick controller to control the ROV movements and functions in the virtual environment. The ROV's controls are scripted and linked to the joystick via a Logitech Profiler. The various buttons and corresponding axis on the joystick are shown in **Table 1**.

3.5. User interface

The user interface (UI) is widely used in many control systems. It is designed to simplify the complicated tasks for the pilot to perform the functions. A well-designed UI can significantly reduce the cognitive load on the end user by merely displaying parameters and having a few buttons. Hence, it allows the user to control the system with focus on essential details [13]. The

Figure 5. TRV-M ROV.

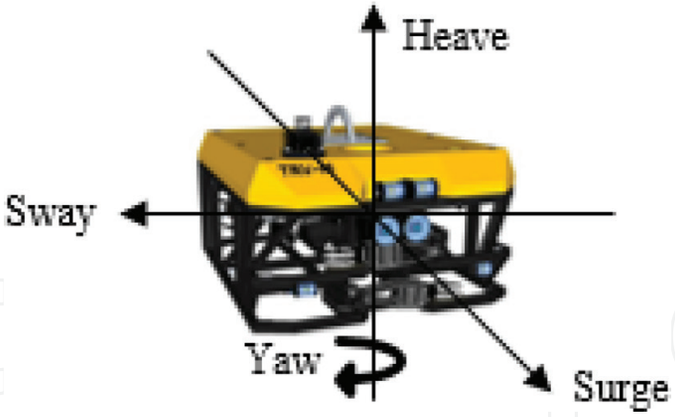

Figure 6. Model of TRV-M ROV with four degrees of freedom used in virtual environment.

Figure 7. Logitech extreme 3D Pro joystick.

ROV function	Control key	
	Keyboard	**Joystick**
Move forward	W	Y-axis
Move backward	S	
Move left	A	X-axis
Move right	D	
Move down	Z	Button 11
Move up	X	Button 12
Yaw left	Q	Twist-axis
Yaw right	E	

ROV function	Control key	
	Keyboard	Joystick
Pitch up	I	POV-north
Pitch down	K	POV-south
Roll left	J	POV-west
Roll right	L	POV-east
Light	F	Button 9
Switch camera view	C	Trigger
Switch between modes	R	Button 3
Manipulator arms activate/deactivate	Tab	Button 2

Table 1. Player inputs for simulator.

Figure 8. User-interface parameters displayed.

UI designed in this project was created using a GameObject called Canvas, in Unity3D. The Canvas encompasses other objects within the User Interface (UI). **Figure 8** shows the UI displaying parameters such as latitude, longitude, depth, heading, speed, run time and small main menu to allow the pilot to restart the practice session.

4. Simulation results with understanding on maritime standards

By the classification standards in the maritime industry, all virtual simulators are assigned with a "Class" namely: Class A, B, C or Class S [14]. The class of a simulator is assigned based on the requirements of the simulator on the checklist provided by the classification society. The comparisons will be made with the standards proposed by DNV (Det Norske Veritas). The comparisons are not meant to be exhaustive but should provide adequate information for designing the virtual simulator.

From the results presented by the project, according to the standards of the classification society DNV, it can be seen that there are some criteria that the proposed Unity3D simulator could not meet. As seen in **Table 2**, items 1.1.19, 1.1.20 and 1.1.21 or any items related to the instructor, these requirements were not met because of the simulator was developed for a single purpose to train the ROV pilot in detecting a leak on a pipeline inspection. Therefore, it did not take into account an instructor to test the competency of the ROV pilot. Another observation in **Table 3** on the behavioral realism for items 2.1.1, 2.1.2, 2.1.5, 2.1.7 and 2.1.8, the class requirements were not fulfilled as the simulation does not encompass an underwater current acting on the objects in the scene. Simulation of actual underwater currents will affect the virtual objects and ROV to make the simulator more realistic. But one would require real information or data to implement it. It was not performed in this chapter. In **Table 4**, items 3.1.2, 3.1.4, 3.1.6 and 3.1.8 under the operating environment require a basic version of turbidity, sea state, underwater fog and camera in the virtual scene. However, some items were included such as camera and underwater lighting in the scene as shown in **Figure 9**. Others were not added due to the lack of actual specifications of these items.

Lastly, as seen in **Table 5**, casualty simulation was not met. The simulation did not have casualty simulation for the ROV. It did not include other equipment malfunctions besides the leakage that occurred in one of the pipelines (see **Figure 10**). In summary, the primary simulator produced has shown that Unity3D possesses the good capability to develop a more realistic virtual simulation for training purposes. Further developments of the simulator were performed to meet these class requirements. For example, a pathfinding algorithm (see **Figure 11**) was used to simulate the autonomous mode of the ROV.

In the virtual simulation, the pathfinding algorithm helps to build the autonomous mode feature in the ROV. Several waypoints were placed in the scene to better guide the ROV along its inspection path. However, using this method, the ROV would pass through the terrain to reach its next point. Few blocks were placed within the terrain to act as obstacles along the pathfinding. They appeared invisible in the simulator as their mesh renders can be switched off. However, the proposed guidance system would only be applicable if the simulated ROV carries out its autonomous functions at a fixed depth. A comprehensive way to create an autonomous guidance system for the ROV was to use a NavMesh algorithm as implemented in **Figure 11**. Any scene can easily be navigated by a GameObject with a NavMesh agent component. The NavMesh allows the simulated ROV to detect the size and height of objects in the scene that cannot pass through. The UI can be improved through the implementation of the User Interface where control buttons can be placed on the screen to allow the pilot to toggle them easily. The buttons on the GUI will serve to remind the pilot if a specific function is switched "on" or "off". As shown in **Figure 9**, the intended ROV GUI for the simulator with a button in the scene to toggle between the multiple camera views and front light on the ROV was implemented. As mentioned, the simulation of underwater currents will help the simulation to look more realistic for the pilot. With the physics supported by Unity3D, the ROV in the simulation can be subjected to these forces and moments caused by the underwater current. It will be implemented in the future works.

The simulation test results will focus on pipeline inspection tasks near to the seabed. The different parts of developing the simulation are integrated together with a scenario set up where there is a gas leakage in one of the jumpers in the top section of the subsea system. First, the ROV's

C1 physical realism		
Item	**Requirement**	**Fulfilled**
1.1.1	Displays and control configurations should be organized similarly to an actual ROV console.	No
ROV model		
1.1.2	ROV models available in the simulation should be similar to the ROV in real life.	Yes
1.1.3	An altimeter should be present to show the height of the ROV above objects.	No
1.1.4	The ROV should have a minimum of one manipulator. The manipulator shall have a similar outlook and behavior to that of it in real life.	Yes
Monitoring		
1.1.5	The simulated control panel shall have a similar outlook and encompass the information required by the pilot to operate the ROV.	Yes
1.1.6	Digital representations of the simulated ROVs thrust, lighting, depth and heading, date and time.	Yes
1.1.7	The digital sonar shall encompass a display window for navigation purposes as well as survey missions. It shall also provide detailed sonar images in consideration to the ROV's position and heading.	No
1.1.8	The tether to the ROV, if there is one, should be dynamic with a display showing the tension and length of the tether.	No
1.1.9	Navigational information should encompass depth, altitude, speed and heading of the ROV.	Yes
Control		
1.1.10	Control over thrust of ROV, hence allowing the pilot complete control of thruster.	Yes
1.1.11	Navigational control over the ROV's movement should be similar to those used for actual ROVs.	Yes
1.1.12	Individual manipulators should be manipulated via hardware controls.	Yes
1.1.13	The pilot should be able to switch between the various camera views available.	Yes
1.1.14	All panning and tilting units should be controllable by the pilot.	No
1.1.15	The pilot should be given a control function to dim the appropriate lights.	Yes
1.1.16	The pilot should be able to winch the tether fully at will.	No
1.1.17	An active or passive motion compensator.	No
1.1.18	A Launch and Recovery System (LRS) for Transportation Management System (TMS) operations, facilitating the docking and release of the ROV.	No
Instructor station		
1.1.19	A separate station with a 3D view should be implemented for the instructor.	No
1.1.20	Underwater visibility should be controllable by the instructor.	No
1.1.21	Underwater currents should be controllable by the instructor.	No

Table 2. Classification standards for physical realism.

task was to identify the location of the leakage, and second, the ROV will need to deactivate the whole system by pressing the switch on the subsea system. The data for the search operation were exported and plotted into a graph, and a video of the simulation was recorded.

C2 behavioral realism		
Item	**Requirement**	**Fulfilled**
2.1.1	A rigid body with 6 DOF (Degree Of Freedom) should be present in the ROV and have the ability resolve the various forces on objects within the simulation.	No
2.1.2	The simulated ROV should be able to resolve moments and forces acting on the ROV.	No
2.1.3	Deployment of the ROV from a vessel which follows the ROV on its own should be included.	No
2.1.4	Exercise areas, along with the various landmass, visuals, buoys tides, and depth data should be available when required to attain the desired training outcome.	No
2.1.5	The simulation should encompass multiple scenarios in dynamic environments with complete object interactions, using detailed dynamics, mechanisms and colliders.	No
2.1.6	The Sonar simulated and data attained from said simulated sonar should be similar to a sonar commonly used in real life.	No
2.1.7	It should be possible to simulate the effects of heave on various mechanical parts.	No
2.1.8	The ROV's tether should be responsive to the simulated environment.	No

Table 3. Classification standards for behavioral realism.

C3 operating environment		
Item	**Requirement**	**Fulfilled**
Visuals		
3.1.1	All visuals should have an engaging 3D environment with detail visuals.	Yes
3.1.2	Underwater visibility should controllable.	No
3.1.3	Lights should be adjustable by the pilot.	Yes
3.1.4	Pilot should have control over cameras allowing him to zoom, adjust and focus.	No
Environmental		
3.1.5	It should be possible to control the surge and speed of currents.	No
3.1.6	It should be possible to control sea state.	No
3.1.7	Interaction with sea floor should be realistic within simulation, e.g. clouding as a result of thrusters or suction.	No
3.1.8	Turbidity of the water should be controllable.	No

Table 4. Classification standards for operating environment.

As seen in **Figures 12** and **13**, the ROV starts at position (0, −793, 0 m), which is at the center of the subsea production layout. The ROV then rotates about its Y-axis in an anti-clockwise direction shown in **Figure 13**, where the ROV's heading is facing approximately 190° away from its global coordinate before it starts moving in X and Z-direction again. As shown in **Figure 14**, the ROV moves along at a constant depth of −793 m for the first 38 s before it moves up as the vehicle is too close to the seabed (to avoid collision). In **Figures 15** and **16**, the forward and lateral thrust (in N) produced by the ROV increases to reach the targeted location, i.e., to maneuver to the desired position. The thrusts reduce once the gas leakage

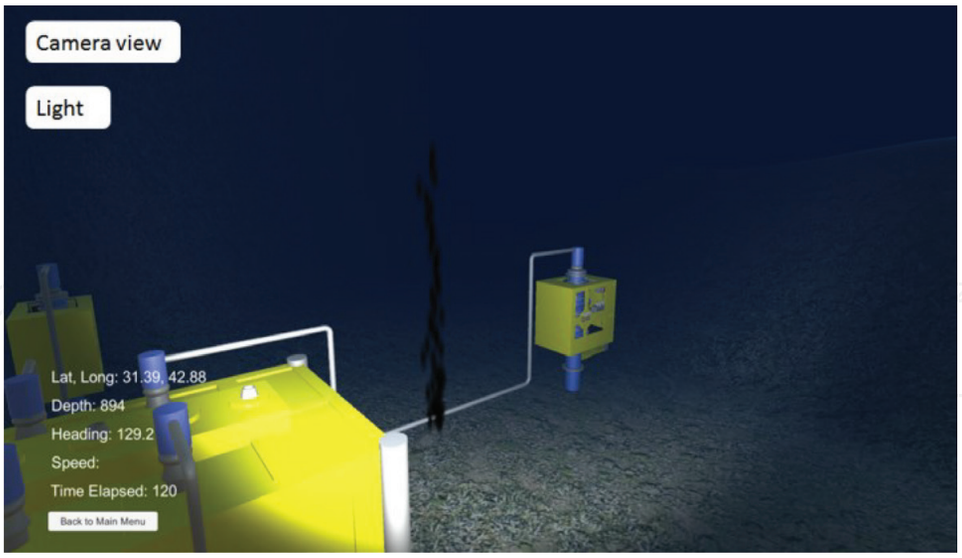

Figure 9. GUI with buttons to toggle between cameras and lighting.

C4 casualty simulation		
Item	**Requirement**	**Fulfilled**
4.1.1	It should have the capability to simulate impairment caused by severe strain on umbilical cord.	No
4.1.2	It should have the capability to simulate damages due to collision impact.	No
4.1.3	It should have the capability for the instructor to inject video display complications.	No
4.1.4	It should have the capability for the instructor to inject thruster control complications.	No
4.1.5	It should have the capability for the instructor to inject instrument display complications.	No
4.1.6	It should have the capability for the instructor to inject sonar display complications.	No

Table 5. Classification standards for casualty simulation.

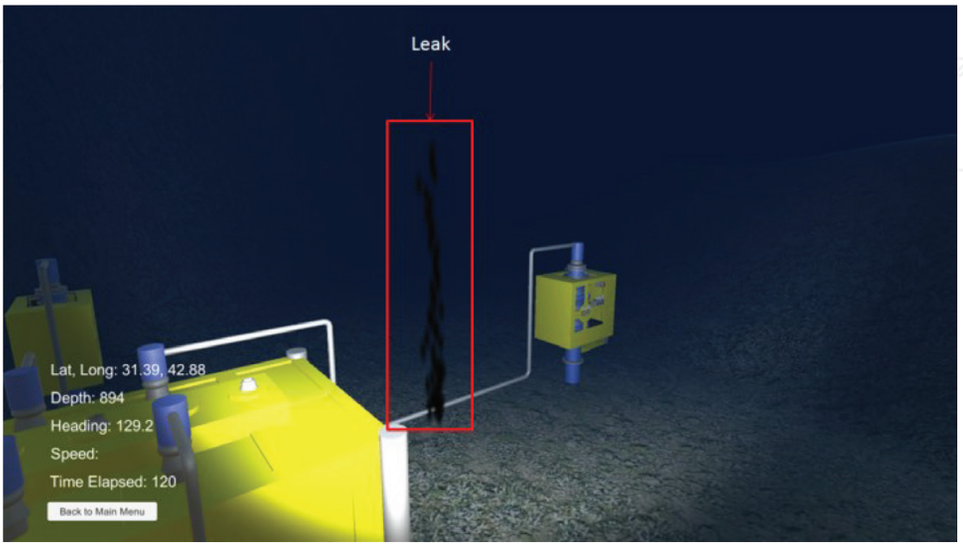

Figure 10. Leakage in pipeline.

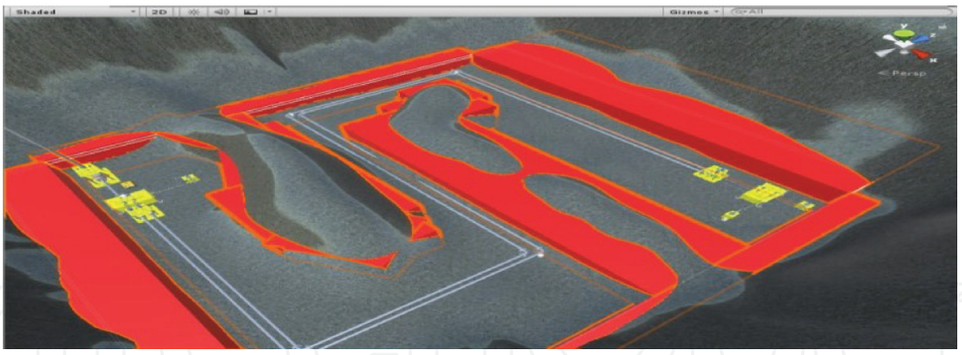

Figure 11. Obstacles in the virtual scene.

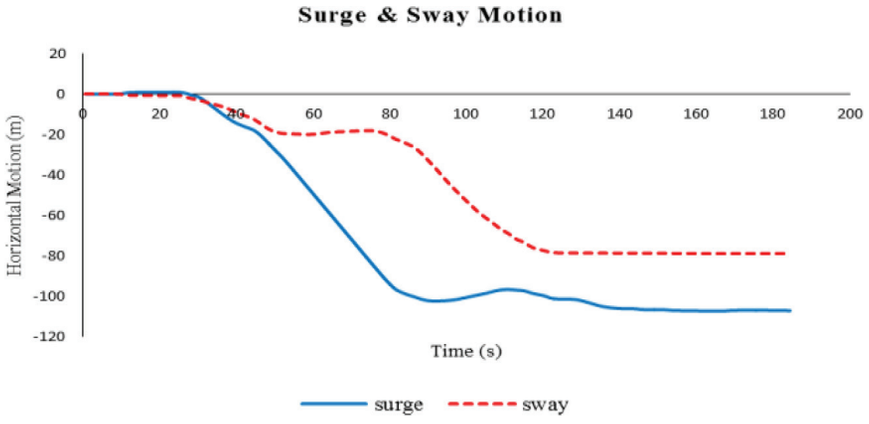

Figure 12. X and Y positions of ROV at the different time frame.

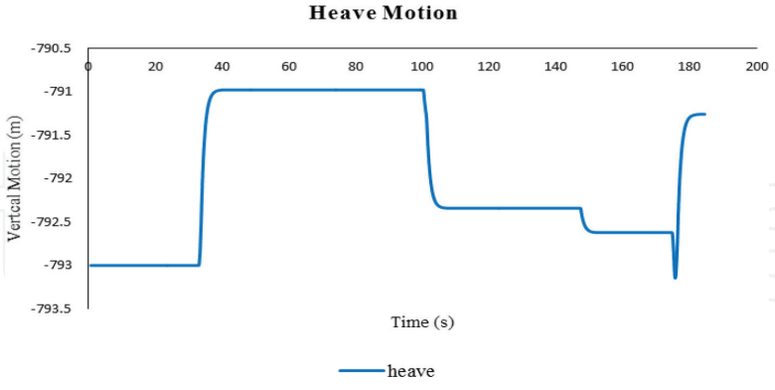

Figure 13. Z position of ROV at various time frames.

and subsea system are found. **Figures 12–14** show less fluctuating in the motion along the X, Z-axis, and yaw. The position of the ROV increases along the Z-axis while its position along the X-axis remains quite constant. The ROV makes a turn around the Y-axis (clockwise) to approximately 280° at 87 s where the position of the ROV increases along the X-axis and remain quite constant on the Z-axis. After approximately 140 s later, the ROV has reached the top section of the subsea system where the gas leakage is located in **Figure 10**.

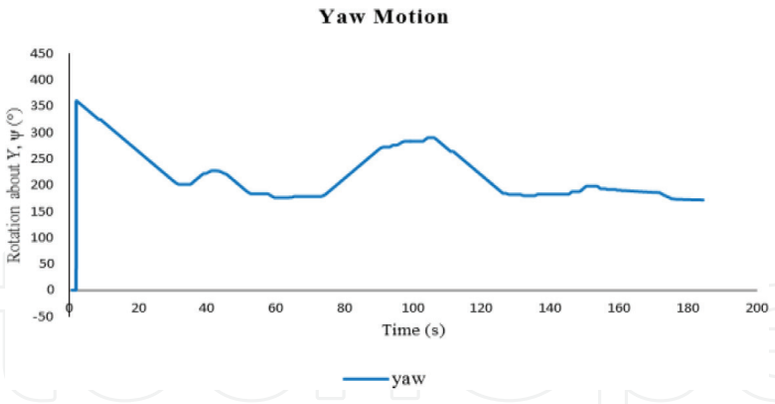

Figure 14. Yaw of ROV at various time frames.

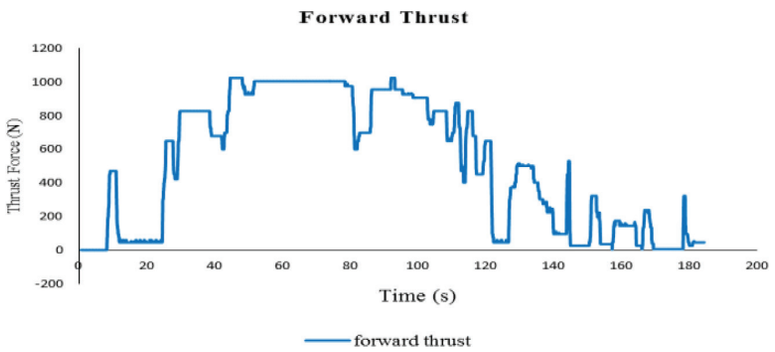

Figure 15. Forward thrust at different time frame.

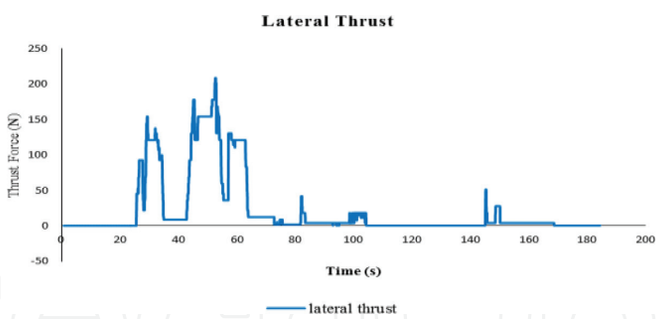

Figure 16. Lateral thrust at different time frame.

The left manipulator of the ROV was activated using a button on the GUI control panel to shut down the subsea system due to the leakage. The ROV can reach the targeted position and shut down system successfully after about 202 s. With the presence of the obstacles in the scene, the objects can interact with one another as though in the real environment. There was no obstacle being hit. The ROV pilot managed to find the gas leakage before reaching the targeted top section of the subsea system to deactivate the switch to prevent more leakages from happening. A sample of the simulation results obtained from the ROV simulator is summarized in **Table 6**.

Descriptions	Results
Total distance traveled in x direction	79.29 m
Total distance traveled in z direction	107.27 m
Final position on y-axis	−792.57 m
Final ROV is heading	169.75°
Total time was taken	202.18 s
Obstacle hits	0

Table 6. Simulated results obtained from ROV pilot simulator.

5. Conclusions

This chapter has successfully demonstrated the process of developing a virtual simulator using an open-source modern game engine software to develop "serious game" as a training system. By integrating different game objects using event-driven programming that is provided within the development software, it can be seen that modern game engine is capable of producing an appropriate level of accuracy in a fraction amount of time and at a low cost. Pilot training is useful for search and rescue operation, dynamic positioning of the drill string, subsea operation, and inspection tasks. Working at the simulator console, the trainees learn necessary flying skills and experience challenges of offshore operations before their first jobs. The virtual ROV's simulator creates a powerful and an efficient project preview and assessment tool for training. The low-cost ROV's pilot simulator can simulate live inputs from the vessel, rig to position the drill string and co-ordinates work operations efficiently in an offshore environment. It dramatically enhances the current training capability, workplace safety in an uncertain environment and reduces maintenance cost as the present cost of a complete ROV's pilot simulator is quite expensive to use and maintain.

Further research can be made to examine whether a pilot candidate who experienced the training in a simulated environment will perform better than those who do not have prior training. The design of different controllers can be implemented to add robustness to the ROV operation in underwater. Further development could be shown to improve on the simulator to suit the requirements of the classification society and specifications. It will be necessary as ROVs are used in many underwater tasks with more autonomy in its operations. However, it is still not possible to replace the decision-making process of a human operator in the loop.

Acknowledgements

The authors would like to thank Newcastle University for supporting the project.

Author details

Cheng Siong Chin[1]*, Xionghu Zhong[2], Rongxin Cui[3], Chenguang Yang[4] and Mohan Venkateshkumar[5]

*Address all correspondence to: cheng.chin@ncl.ac.uk

1 Newcastle University, Singapore, Singapore

2 Visenti Pte Ltd, Singapore, Singapore

3 School of Electronic and Control Engineering, Northwestern Polytechnical University, Chang'an University, Xi'an, China

4 Zienkiewicz Centre for Computational Engineering, Swansea University, Swansea, UK

5 Department of EEE, Aarupadai Veedu Institute of Technology, Chennai, Tamil Nadu, India

References

[1] Christ R, Wernli SR. The ROV Manual, Second Edition: A User Guide for Remotely Operated Vehicles. Second ed. Waltham, USA: Elsevier; 2014

[2] Chin CS, Lau MWS, Low E. Supervisory cascaded controllers design: Experiment test on a remotely-operated vehicle. Proceedings of the Institution of Mechanical Engineers: Journal of Mechanical Engineering Science. 2010;**225**(3):584-603

[3] Chin CS, Lau MWS, Low E, Seet GG. Software for modelling and simulation of a remotely operated vehicle. International Journal of Simulation Modeling. 2006;**5**(3):114-125

[4] Cui R, Chen L, Yang C, Chen M. Extended state observer-based integral sliding mode control for an underwater robot with unknown disturbances and uncertain nonlinearities. IEEE Transactions on Industrial Electronics. 2017;**64**(8):6785-6795

[5] Cui R, Yang C, Li Y, Sharma S. Adaptive neural network control of AUVs with control input nonlinearities using reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2017;**47**(6):1019-1029

[6] Schjolberg I, Utne IB. Towards autonomy in ROV operations. IFAC-PapersOnLine. 2015; **48**(2):183-188

[7] Chin CS, Lin WP. Robust genetic algorithm and fuzzy inference mechanism embedded in sliding-mode controller for uncertain underwater robot. IEEE/ASME Transactions on Mechatronics. 2018;**32**(2):655-666

[8] Xie J. Research on key technologies base Unity3D game engine. In: 7th International Conference on Computer Science & Education (ICCSE); Melbourne, Australia: 2012. pp. 695-699

[9] Chin CS, Kamsani NB, Zhong XH, Cui R, Yang C. Unity3D serious game engine for high fidelity virtual reality training of remotely-operated vehicle pilot. In: 10th International Conference on Modelling, Identification and Control, Guiyang, China; 2-4 July 2018

[10] Bartneck C, Soucy M, Fleuret K, Sandoval EB. The robot engine—Making The Unity 3D game engine work for HRI. In: Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN2015), Kobe; 2015. pp. 431-437

[11] Altabel, Altabel Group's Blog Unreal Engine 4, Unity, Cry Engine: What to choose? Available from: https://altabel.wordpress.com/2015/01/22/unreal-engine-4-unity-cry-engine-what-to-choose/ [Assessed: 14-05-2018]

[12] Juarez A, Schonenberg W, Bartneck C. Implementing a low-cost CAVE system using the CryEngine2. Entertainment Computing. 2010;**1**(3-4):157-164

[13] Candeloro M, Sorensen AJ, Longhi S, Dukan F. Observers for dynamic positioning of ROVs with experimental results. IFAC Proceedings Volumes. 2012;**45**(27):85-90

[14] Veritas DN. Standard for Certification No. 2.14 Maritime Simulator Systems. Det Norske Veritas (DNV) Standards for Certification, Høvik, Norway. 2011. pp. 75-78

IntechOpen