Summer 2017

Design of A Virtual Laboratory for Automation Control

Zelin Zhu
Old Dominion University

DESIGN OF A VIRTUAL LABORATORY FOR AUTOMATION CONTROL

by

Zelin Zhu
B.S. May 2012, Nanjing University of Posts and Telecommunications

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

MODELING AND SIMULATION

OLD DOMINION UNIVERSITY
August 2017

ABSTRACT

DESIGN OF A VIRTUAL LABORATORY FOR AUTOMATION CONTROL

Zelin Zhu
Old Dominion University, 2017
Director: Yuzhong Shen

In the past, only students who studied on campus were able to access laboratory equipment in traditional lab courses; distance learning students, enrolled in online courses, were at a disadvantage for they could learn basic lab experiment principles but could never experience hands-on learning. Modeling and simulation can be a powerful tool for generating virtual laboratories for distance learning students. This thesis describes the design and development of a virtual laboratory for automation control using mechanical, electrical, and pneumatic components for an automation and control course at Old Dominion University. This virtual laboratory application was implemented for two platforms — Windows personal computers and Android smartphones. The virtual lab serves as pre-lab session for on-campus students and a virtual lab tool for distance-learning students to gain some “hands-on” lab experience. Utilizing the virtual learning environment as a supplement to engineering-based laboratories is also beneficial for students to prepare for the physical experiment and obtain a “hands-on,” practical lab experience without the hazards present in the physical lab. Such a methodology can also be applied to experiments in different fields such chemistry, etc.

This thesis is dedicated to my beloved family and friends, who offered unconditional love and support and have always been there for me.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my advisor Dr. Yuzhong Shen for his constant guidance, support, and encouragement throughout my graduate program of study. I also want to thank my committee members, Dr. Zhanping Liu and Dr. Frederic McKenzie, for their valuable comments and suggestions on my research. I would like to acknowledge Dr. Cheng Lin for the research data. I also would like to thank Shuo Ren for his expertise on interface and layout design for my project. Special thanks go to Alexandra Rich and Chongyuan Du, who offered kind support and expertise for my thesis revision and UI icon design. Last, but not the least, I would like to thank my family for their continuous support with love and care.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Education and learning have become easier and more enjoyable with advances in computer software and hardware. Nowadays, there are other methods to learn new knowledge and skills outside the traditional classroom environment. In May 2012, Massachusetts Institute of Technology [1] and Harvard University [2] founded edX [3], which is a Massive Open Online Course (MOOC) learning platform that runs on an open-source software platform (Open edX) [3]. It offers online university-level courses in a wide range of disciplines to all the students around the world. Although traditional classroom instruction still plays arguably the most important role in teaching and learning, students and professionals have much more resources at their disposal to expand their knowledge and develop expertise through massive open online courses from the world's best universities. Many schools, nonprofit organizations, and corporations offer or will soon offer courses on the edX website[1].

In addition to online courses, web search engines can also provide answers for all kinds of questions with a single mouse click. Free video websites like YouTube offer a myriad of high quality video tutorials on many subjects, ranging from mathematics to physics to computer science and others. Students and professionals can also obtain various novel ideas by listening to experts on online educational channel such as TED Talks [4]. With the ubiquity of smartphones and portable tablet computers, scores of mobile applications have been developed for learning and education. The ability for self-

[1] IEEE Transactions and Journals style is used in this thesis for formatting figures, tables, and references.

study, anywhere and anytime, is increasing and in strong demand among students and professionals alike.

However, while online education does provide many new opportunities and resources, one limitation of existing online education is that it cannot provide learners with practical know-how or “field work.” Instead, online educational resources often focus on teaching theoretical concepts, such as proving theorems or solving equations, and do not offer any laboratory exercises. Yet, laboratory exercises are an essential component of science and engineering education, as they provide hands-on experiences for understanding theoretical concepts or directly addressing real world problems.

There are several approaches to address the lack of hands-on laboratory exercises in online education. The first approach is to provide recorded videos of real lab experiments to familiarize students with the process of how a certain lab is conducted. However, this method alone does not adequately substitute for the physical experiment, as learners have no opportunity or lab equipment to practice the skills they learned from the video. The second approach, which holds far greater potential to enhance the online laboratory experience, is to use computer-based modeling and simulation (M&S) applications. In highly realistic 3D simulated environments, a virtual laboratory simulation can emulate lab equipment and configurations and provide a comparable “hands on” learning experiences.

Computer-based modeling and simulations that use computer hardware and software to model and simulate a wide range of problems have been successfully applied in many fields, particularly in science and engineering fields. For example, medical students and professionals can safely perform surgeries on virtual patients in a simulated environment to enhance their surgical skills [5-7]. Unlike most traditional laboratory sessions where students are typically required to prepare experiments by reading and studying the paper-based lab manual, pre-lab sessions in a computer-based learning environment offer students unique learning experiences and practical activities, while also helping them understand the experimental objectives and procedures in a more interactive way prior to the physical experiment [8-10]. Moreover, using M&S tools help to avoid damages to physical equipment and reduce common mistakes so that the learners can perform more efficiently during the physical experiment by avoiding the mistakes made in the virtual laboratory. The rich and realistic visual effects of the virtual learning environment also make the pre-lab session more immersive and engaging.

Electrical and computer engineering is a field in which modeling and simulation are extensively used. Before circuit simulation tools were invented, electrical engineers had to manufacture and examine a real circuit board or integrated chip to check whether the circuit design was correct or not. In order to reduce the cost and time of circuit design, various circuit simulations languages and tools have been developed, such as SPICE [11] for analog circuit design and VHDL [11] for digital circuit design. These circuit simulation tools generate highly accurate results, and new designs can be simulated and tested using simulation before putting them into production, greatly reducing the cost of circuit design.

For introductory courses in circuit design and analysis, breadboards are widely used for building relatively simple circuits and learning basic circuit principles. However, students often don’t have enough exposure to laboratories because of the limited lab resources, such as available space, equipment, and scheduling. The problem becomes much more severe with online education and/or distance learning because it is almost impossible for online/distance learning students to access lab resources. Thus, there is an urgent need to provide alternate means to complement labs for both traditional and online education. In recent years, there have been a few released applications attempting to build virtual circuits. However, many of them are 2D-image or symbolic simulation. Wiring is a very critical procedure for breadboard circuits and the reduction of one spatial dimension of using 2D images for 3D wires causes disconnections between the virtual breadboards and real ones and reduces the effectiveness of virtual breadboard simulation.

This thesis describes the software design and development of a highly realistic virtual lab for learning programmable control logic in an automation and control lab. In particular, this thesis presents the design and development of a simulation tool, named Virtual Lab, for wiring programmable control logic in an automation and control course (MET 370/386 Automation and Control) offered at Old Dominion University. Virtual Lab provides distance-learning students with a realistic lab scenario and accurate simulation of Programmable Logic Controllers (PLCs) equipment. Before using Virtual Lab, students can only use Computer Aided Design (CAD) software to complete their homework and lab problems without being able to touch PLCs or wire ladder logic diagrams [12]. Virtual Lab contains mechanical, electrical, and pneumatic components that can be combined and wired to form various control logic.

Virtual Lab fills a gap in existing circuit and control simulation software, which has nearly 40 years of history with mature industry standards and is widely used in design of circuit and control systems. Existing circuit and control simulation software is designed for professionals and its visual presentations are symbolic and restricted to circuit or system levels. The major contribution of Virtual Lab is that it provides a three-dimensional realistic lab environment for electro-pneumatic experiments with an intuitive and user-friendly graphical interface. It also provides a highly interactive virtual environment for the learners to conduct the experiment with ease. The core of Virtual Lab is a programmable logic control simulation engine that supports mechanical, electric, and pneumatic components and it allows combination of these different types of components for programmable logic control for automation.

The remainder of this thesis is organized as follows: Section 2 describes background and related work. Section 3 presents a previous implementation of Virtual Lab that is expanded in this thesis. Section 4 shows the current methodology and implementation of Virtual Lab. Section 5 presents the validation results of Virtual Lab. Section 6 discuss future work and concludes the thesis.

CHAPTER 2

BACKGROUND

This section discusses background and related work. It starts with a brief introduction to modeling and simulation, followed by an overview of applications of modeling and simulation for education. The last section describes the physical lab components to be modeled and simulated in Virtual Lab.

2.1 Modeling and Simulation

Modeling and simulation is the process of creating a system or phenomenon by using physical, mathematical, or logical methods. Modeling and simulation facilitates analysis, experimentation, and training. In this way, researchers can better understand a system's behavior without testing the system in the physical world.

The first use of M&S can be traced back to Buffon's "needle experiment" in 1777, in which the value of "π" can be estimated by dropping needles on a floor marked with equally spaced parallel lines [13]. After approximately a century and a half later, M&S was used in training pilots during the first World War. The Antoinette Trainer [14], consisting of two half section barrels mounted on top of each other, was the first flight simulator built by a French manufacturer to simulate the pitch and roll control of a monoplane. In the late 40s and early 50s, commercially designed computers started to arise as problem solving tools in many different organizations worldwide [15]. With the maturation of computer technology, M&S continued to evolve as popular training tools in the period of 50s – 80s. Multi-user synthetic environments were used to train soldiers under different mission and combat scenarios.

Recent technological advancements in M&S have enabled the creation of realistic virtual environments. For example, M&S is routinely used by the military for recruitment and training and for development of tactical and strategical solutions. On Air Force flight simulators, controls, fittings, and other elements are in the same places as they would be on a real aircraft. In fact, flight simulators have been used to train pilots across all branches of the forces to great success. Furthermore, M&S can be used in combat simulation as well. Soldiers and other related personnel can utilize virtual reality glasses that create an illusion of three-dimensional depth, and the results of this can be shared amongst large numbers of personnel [16, 17]. In addition to the military, modeling and simulation are widely used in other fields as well, including but not limited to, science, engineering, healthcare, transportation, and education.

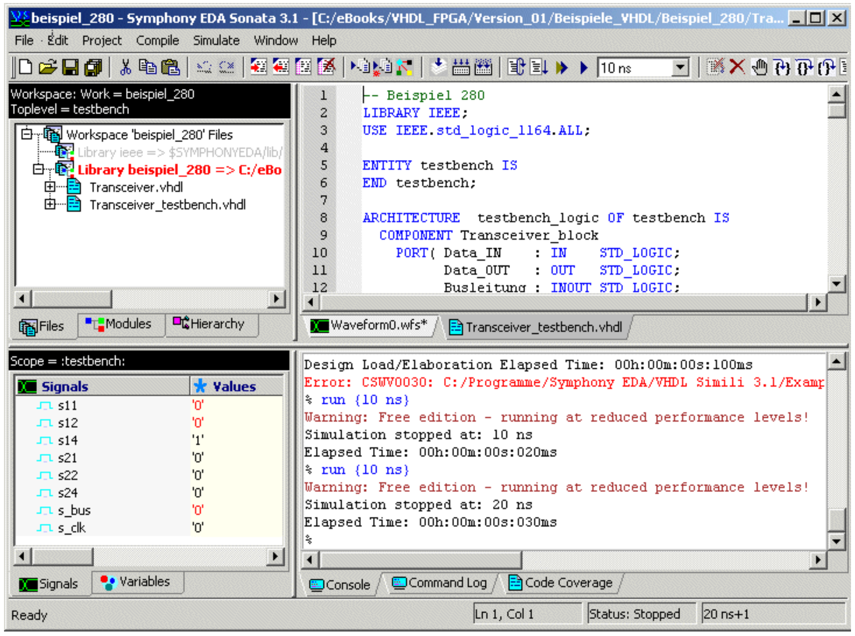
2.2 Modeling and Simulation for Education

As previously posited, M&S can help researchers understand a system's behavior without testing said system in the physical world. M&S is utilized as a learning tool in the form of educational applications. M&S software facilitates studying material, exploring or expanding concepts, and acquiring certain skills in a virtual-based structure. With improvements in both software and hardware in the past two decades, M&S has become more popular and widely used in the academic world. A well-designed education application often includes: 1) multiple levels or challenges, 2) a compelling or intriguing guideline, 3) an intuitive, unique teaching method for every learner, 4) additional feedback from institutions to the designers.

Educational computer applications and simulation are often referred to as Game-based learning (GBL) tools. GBL provides an approach of learning in which users explore and interact with educational content in an entertaining format [14]. Within an effective game-based learning environment, students and professionals work toward certain goals, choosing actions and experiencing the consequences of those actions along the way. Mistakes are made in a risk-free setting, and users safely learn and practice the correct protocol through many experiments. GBL keeps learners highly engaged in practicing behaviors and thought processes that can be easily transferred from the simulated environment to real life.

Before 3D computer graphics were widely available to educational applications, 2D computer graphics were used to present two-dimensional geometric models. They were used in applications that were originally developed for the traditional printing and drawing medium. Several commonly used engineering modeling and simulation tools mostly used 2D computer graphics, such as VHSIC Hardware Description Language (VHDL), Simulation Program with Integrated Circuit Emphasis (Spice), LabVIEW, and MATLAB.

VHDL is a hardware description language to model digital and mixed-signal systems mainly in electronic design applications. It is commonly used to generate models that describe a logic circuit. The logic design of that logic circuit is processed by a synthesis program and tested by a simulation program using simulation models. This collection of simulation models is commonly called a test bench. VHDL also has the capabilities of file input and output, and it can be used as a general-purpose language for text processing, but files are more commonly used by a simulation test bench for stimulus or verification data. There are some VHDL compilers which build executable binaries. In this case, it might be possible to use VHDL to write a test bench to verify the functionality of the design using files on the host computer to define stimuli, to interact with the user, and to compare results with those expected. However, most designers leave this job to the simulator [18]. The example of using VHDL is shown below in Fig. 1.


beispiel_280 - Symphony EDA Sonata 3.1 - [C:/eBooks/VHDL_FPGA/Version_01/Beispiele_VHDL/Beispiel_280/Tra...
File Edit Project Compile Simulate Window Help
10 ns
Workspace: Work = beispiel_280
Toplevel = testbench
Workspace 'beispiel_280' Files
Library ieee => $SYMPHONYEDA/lib/
Library beispiel_280 => C:/eBo
Transceiver.vhdl
Transceiver_testbench.vhdl
Files
Modules
Hierarchy
Waveform0.wfs*
Transceiver_testbench.vhdl
Scope = :testbench:
Signals
Variables
Console
Command Log
Code Coverage
Ready
Ln 1, Col 1
Status: Stopped
20 ns+1

```
-- Beispiel 280
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY testbench IS
END testbench;

ARCHITECTURE  testbench_logic OF testbench IS
  COMPONENT Transceiver_block
    PORT( Data_IN    : IN    STD_LOGIC;
          Data_OUT   : OUT   STD_LOGIC;
          Busleitung : INOUT STD_LOGIC;
```

Signals	Values
s11	'0'
s12	'0'
s14	'1'
s21	'0'
s22	'0'
s24	'0'
s_bus	'0'
s_clk	'0'

```
Design Load/Elaboration Elapsed Time: 00h:00m:00s:100ms
Error: CSWV0030: C:/Programme/Symphony EDA/VHDL Simili 3.1/Examp
% run {10 ns}
Warning: Free edition - running at reduced performance levels!
Simulation stopped at: 10 ns
Elapsed Time: 00h:00m:00s:020ms
% run {10 ns}
Warning: Free edition - running at reduced performance levels!
Simulation stopped at: 20 ns
Elapsed Time: 00h:00m:00s:030ms
%
```

Fig. 1. VHDL Interface [18].

When it is used for systems design, the major advantage of VHDL is that it allows the behavior of the required system to be modeled and verified before synthesis tools translate the design into real hardware. Another benefit of using VHDL is that it allows the description of a concurrent system instead of running codes sequentially, one instruction at a time.

SPICE stands for Simulation Program for Integrated Circuits Emphasis. It was originally developed in 1975 at Electronics Research Laboratory of University of California, Berkeley. It is a powerful circuit simulator that is widely used in integrated circuit and board-level design to verify circuit designs and to predict the circuit behavior. The advantages of using SPICE includes: 1) it is relatively cheap and has several free versions to download; 2) it is commonly used for teaching electronic related courses in most universities; 3) it provides users a very accurate circuit simulator tool. The interface of SPICE is shown below in Fig. 2.

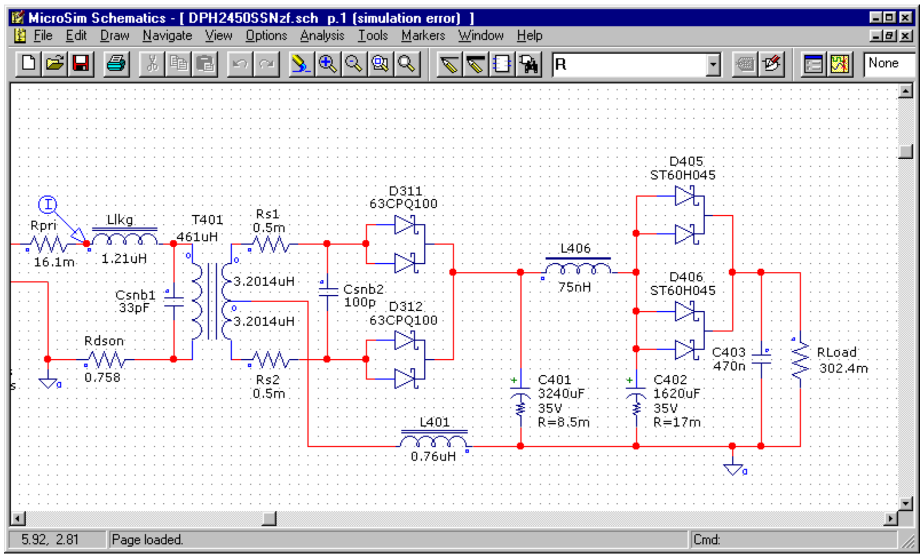


Fig. 2. SPICE Interface [11].

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a system-design platform and development environment using a visual programming language from National Instruments. The programming language used in LabVIEW, named G, is a dataflow programming language. Execution is determined by a graphical block diagram which connects different function-nodes by drawing wires. LabVIEW integrates the creation of user interfaces into the development cycle. LabVIEW programs-subroutines are termed virtual instruments (VIs). Each VI has three components: a block diagram, a front panel, and a connector panel. The front panel is built using controls and indicators. Controls are inputs, which allow a user to supply information to the VI. Indicators are outputs, which can indicate, or display, the results based on the inputs given to the VI. The objects placed on the front panel will appear on the back panel as terminals. The back panel also contains structures and functions which perform operations on controls and supply data to indicators. The structures and functions are found on the Functions palette and can be placed on the back panel. Furthermore, indicators, structures, and functions are referred to as nodes. Nodes are connected to one another using wires, e.g., two controls and an indicator can be wired to the addition function so that the indicator displays the sum of the two controls. Thus, a virtual instrument can be run as either a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program [19]. One example of LabVIEW is shown in Fig. 3.

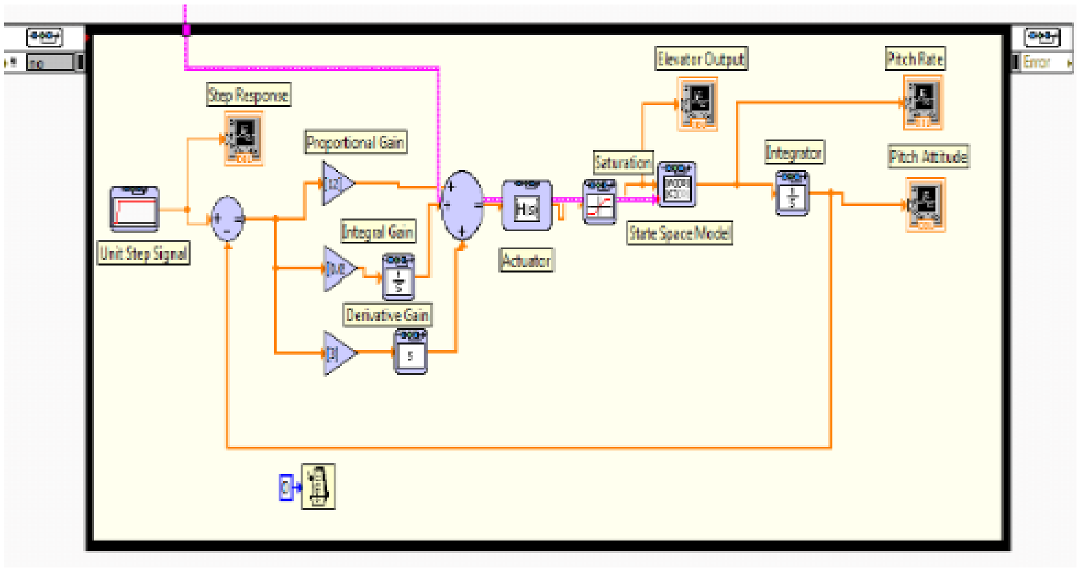


Fig. 3. LabVIEW Interface [19].

When using LabVIEW, the graphical programming environment allows nonprogrammers to build programs by dragging and dropping virtual representations of lab equipment. However, there is also a certain danger of underestimating the expertise needed for high-quality G programming. For complex algorithms or large-scale code, it is significant that a programmer is familiar with LabVIEW code syntax and the topology of its memory management.

MATLAB stands for Matrix Laboratory. It is an interactive software system for numerical computations and graphics. As the name suggests, MATLAB is especially designed for matrix computations, such as solving systems of linear equations, computing eigenvalues and eigenvectors, factoring matrices, etc. In addition, it has a variety of graphical capabilities, and can be extended through programs written in other languages, including C, C++, C#, Java, Fortran and Python [20]. Although MATLAB is primarily designed for numerical computing, it contains an additional toolbox and package that can add graphical multi-domain simulation and model-based design for dynamic and embedded systems. MATLAB applications are created by the MATLAB scripting language. Common usage of the MATLAB application involves using the Command Window as an interactive mathematical shell or executing text files containing MATLAB code. The advantages of using MATLAB includes that: 1) it allows users to interact with their graphical models; 2) it can be greatly expanded by the addition of toolboxes; 3) it has a straightforward interface and quickly comprehends scripting language.

While existing modeling and simulation tools such as VHDL, SPICE, LabVIEW, and MATLAB are very powerful and widely used, one capability that is missing in these tools is realistic rendering of physical objects and environments. For example, only graphical symbols of circuit components are used by both SPICE and VHDL simulation tools, not 3D models of circuit components. Realistic 3D models are needed in various applications, such as circuit diagnosis, and especially useful for novice users. The Virtual Lab developed in this thesis fills a gap in existing simulation tools by providing a realistic rendering of lab components and environments that are very similar to those in the real lab, and thus facilitates learning transfer and improves learning efficacy.

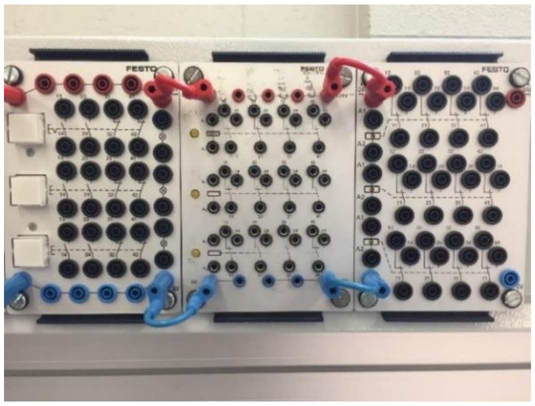
2.3 Lab Description

The Mechanical Engineering Technology Program at Old Dominion University provides a course named MET 370/386 Automation and Control. This course is strongly related to the design and analysis of feedback control system which includes the fundamentals of programmable logic controllers as well as practical applications of interfacing mechanical, electrical, pneumatic and hydraulic feedback control circuits.

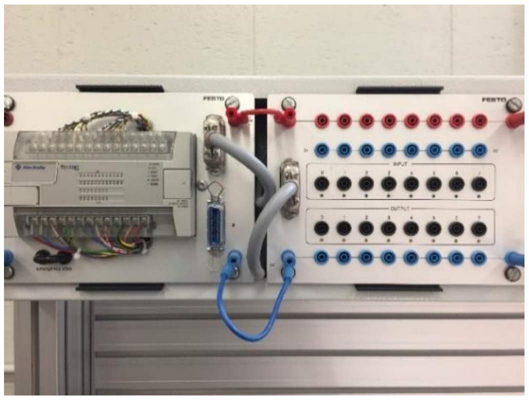
Additionally, it has a lab section which contains: 1) four weeks of pneumatic applications, 2) four weeks of pneumatic components, electrical sensors, and ladder diagram, 3) five weeks of PLC programming [21]. The lab overview is shown in Fig. 4.

Fig. 4. Lab Overview in the Department of Engineering Technology at Old Dominion University [12].

The lab contains five electro-pneumatic training systems for students to conduct physical experiments. Each system consists of three major electrical components (power supply box, manual switch box, and electrical relay switch box) and nine major pneumatic components (air source, single-acting cylinder, double-acting cylinder, air divider, 3/2 pneumatic valve, 3/2 directional control valve, 3/2 solenoid (one sided) control valve, 5/2 solenoid (one sided) control solenoid valve, and 5/2 solenoid control valve), as shown in Fig. 5.



(a)



(b)

Fig. 5. Programmable Logic Controller Wiring Lab. (a) Two major components of the lab (from left to right): manual switch box, and electrical relay switch box. (b) Power supply box.

The power supply provides 24V Direct Current (DV) power. The manual switch box contains three manual push buttons with each controlling four switches: two Normally Open (NO) and two Normally Closed (NC) contacts. TABLE 1 shows the connection states of the switches when the push button is activated or deactivated. The state ON of a switch means that the two nodes of that switch are connected and OFF means those two nodes are disconnected [22].

TABLE 1

SWITCHES AND MANUAL PUSH BUTTON

Connections	Power Status (When Push Button Is Activated)	Power Status (When Push Button Is Deactivated)
13 & 14 (NO)	ON	OFF
23 & 24 (NO)	ON	OFF
31 & 32 (NC)	OFF	ON
41 & 42 (NC)	OFF	ON

Instead of using push buttons, electric relays are used in the electric switch box to control the connection states of the switches. Each relay controls four pairs of NO and NC switches. TABLE 2 shows the connection states of the switches when the relay switch is activated or deactivated [6].

TABLE 2

SWITCHES AND ELECTRICAL RELAY

Connections	Electrical Relay Power Status When Activated	Electrical Relay Power Status When Deactivated
12 & 11(NC)	OFF	ON
22 & 21(NC)	OFF	ON
32 & 31(NC)	OFF	ON
42 & 41(NC)	OFF	ON
14 & 11(NO)	ON	OFF
24 & 21(NO)	ON	OFF
34 & 31(NO)	ON	OFF
43 & 41(NO)	ON	OFF

The air switch is also called directional control valve; it determines which part of the system the air flow enters. It is activated by pressing the green button manually. The button will stay down until it is pressed again, then it will retract to its resting position with the help of a spring. The air switch is useful when the user wants the system to be controlled manually instead of with sensors and other types of initializers. An air switch is illustrated in Fig. 6.

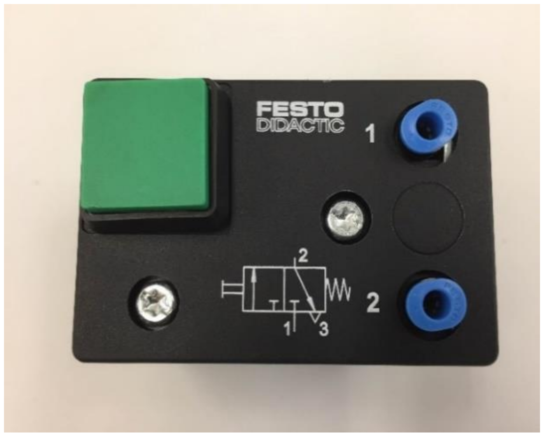
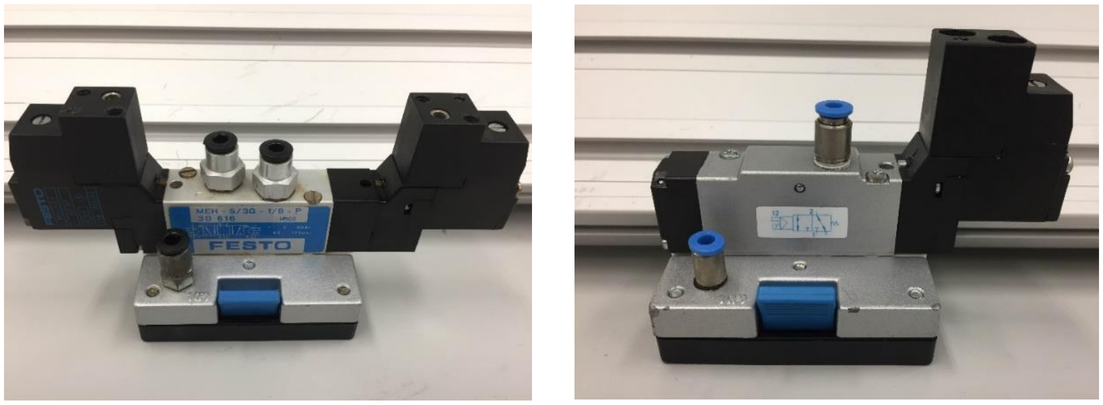


Fig. 6. Air Switch in Pneumatic Lab.

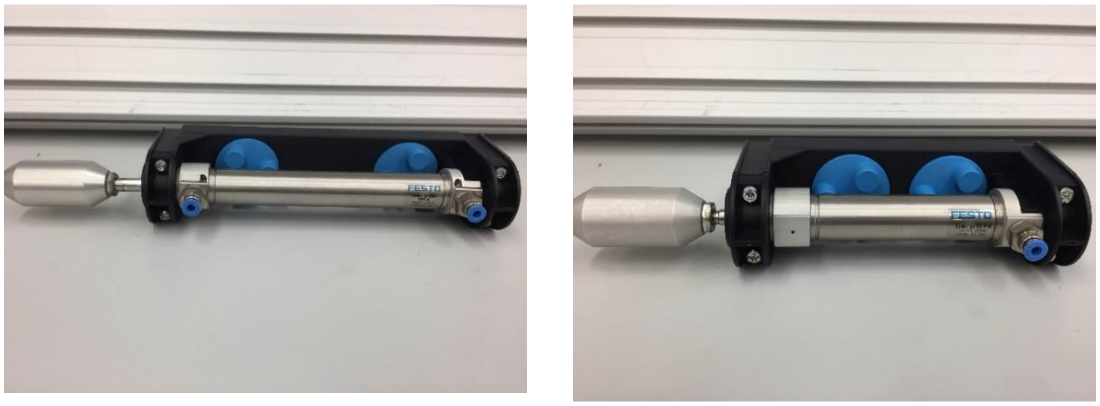
Different than the directional control valves with manual switches, solenoid control valves are activated or deactivated by using electrical signals. The solenoid slots, which are black connectors in Fig. 7, are where the air pipe and pneumatic systems connect. After receiving a signal from a PLC or hardwired system, the valve will be activated and release the air into another part of the equipment. The major difference between 3/2 solenoid control valve and 5/2 solenoid control valve is the number of slots on the equipment. The difference between solenoid one sided control valve and solenoid double sided control valve is that solenoid one sided control valve has a spring inside, which can put it back in resting position after the electrical signal ends. Fig. 7 shows the physical components of the solenoid control valve.



(a) (b)

Fig. 7. Solenoid Control Valve in Pneumatic Lab. (a) 5/2 Solenoid Control Valve. (b) 3/2 Solenoid (One Sided) Control Valve.

Acting cylinder is also named actuator. There are two types of actuators used in high pressure systems. The first is a single-acting cylinder. This type of actuator can be activated by the air being released into its entrance port, which is on the top of the housing of the cylinder. When the air flow stops, the cylinder will move back to its resting position due to an embedded spring. The other type is a double-acting cylinder. Unlike the single-acting cylinder, the double-acting cylinder only retracts to its original position if there is a signal of air entering through another port. Fig. 8 shows two acting cylinders.



(a) (b)

Fig. 8. Acting Cylinders in Pneumatic Lab. (a) Double Cylinder. (b) Single Cylinder.

The air supply device accelerates the air speed by using a high-pressure pump and channeling it through a hose, which sends the air to a pneumatic system. Fig. 9 shows the air supply.

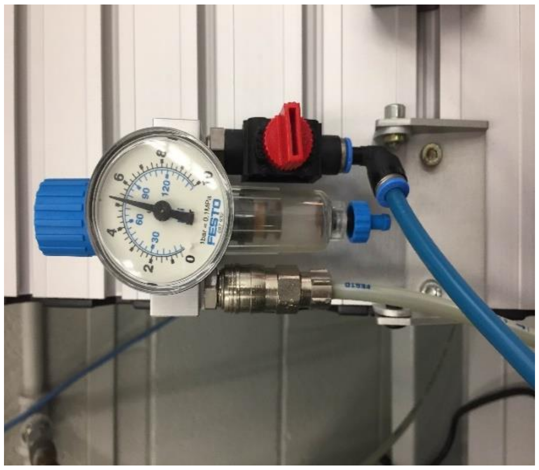


Fig. 9. Air Supply in Pneumatic Lab.

Air dividers can be used to connect the air compressor to various valves and actuators in the system. The only way to control which slot the air exits out is using a hose to connect with that slot. Fig. 10 shows an air divider.

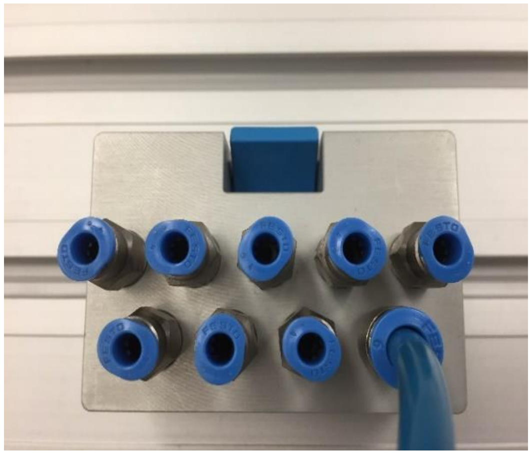


Fig. 10. Air Divider in Pneumatic Lab.

CHAPTER 3

PREVIOUS IMPLEMENTATION

Before Virtual Lab was developed at Old Dominion University, one of the biggest challenges in laboratory-related classes in distance learning programs was to provide hands-on lab experience to those students enrolled in distance learning courses. Thus, on-campus students have more opportunities to access lab equipment and obtain practical lab experience than those students in distance learning programs. To address this problem, the first version of Virtual Lab was designed and developed for wiring labs in an automation and control course.

The main design goal of Virtual Lab was to provide a virtual laboratory environment that is intuitive, interactive, and effective for both learning and teaching PLC wiring. Fig. 11 shows a screen capture of the first version Virtual Lab. All three electrical components were created in 3D using CAD software. Furthermore, the Power Supply Box, Manual Switch Box, and Electrical Switch Box are located at the top, bottom right, and bottom left respectively. Six button icons were also created for various additional functions. From top to bottom, the first icon is called “Cable Color Selection,” and it enables students to select different colors for the wiring cables used in the lab. For example, students can choose red for the cables connected from the +24V power source and blue for the cables connected to the 0V power source. The second icon is “File Save,” and students can use it to save their finished/unfinished project in a binary format file that can be submitted to their instructors or can be opened later to review or modify their unfinished work. The third icon is “File Open.” The instructor can use this icon to open and review a file submitted by students for grading purposes. A student can also use this icon to review or continue working on one unfinished project. The fourth icon is "Volume Control," and students can use this function to control the sound for the node clicking sound or a warning signal, which occurs when a short circuit happens. Next icon is "Help," and after clicking this icon, several PLC programming examples are shown for new users. The last icon is "System Close," and it allows users to close the software safely [6]. Fig. 11 shows the functionality of each icon in Virtual Lab.

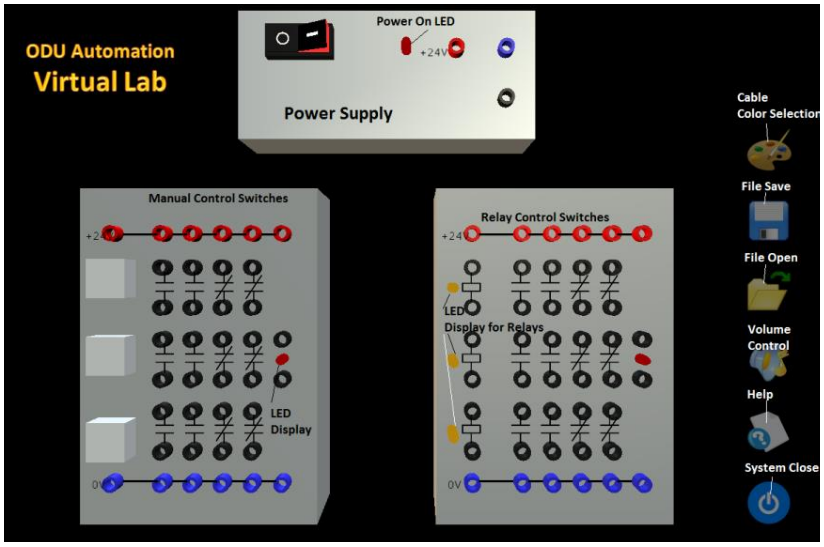


Fig. 11. Functionality of Each Icon in Virtual Lab.

Virtual Lab includes both mechanical and electronical controllers that can be wired to form specific control logic. Virtual Lab was also placed on the website so that students can access the software through PCs, tablets, and smartphones. Fig. 12 shows Virtual Lab running on a Windows 8 tablet.

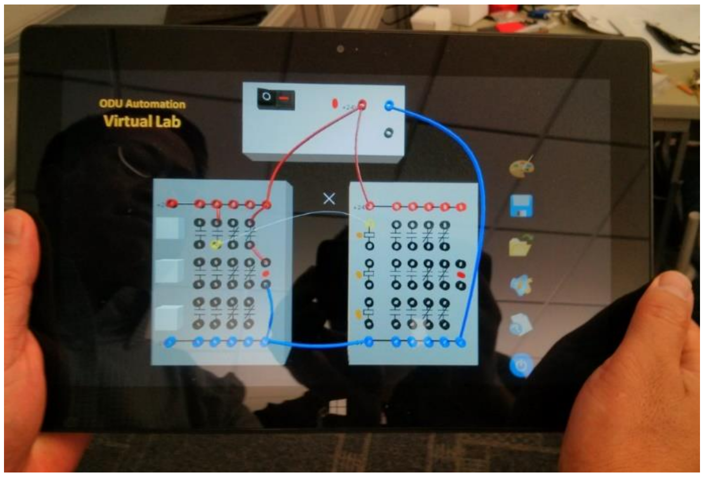


Fig. 12. Virtual Lab running on a Windows 8 Tablet [12].

Virtual Lab was first used in the course MET 370/386 Automation and Control in Fall 2013 in an on-campus class, as there was no distance learning class scheduled at that time. TABLE 3 shows the results of survey questions.

TABLE 3

RESULTS FOR SURVEY QUESTIONS

5: strongly agree; 4: moderately agree; 3: agree; 2: partially agree; 1: disagree completely	
Survey Questions	Results
1. The software helps me gain the knowledge in logic-gate wiring.	4.41
2. The software saves me time to learn logic-gate wiring and improves my actual wiring speed significantly.	3.89
3. The software functions almost the same as the real equipment.	4.00
4. The software is user friendly.	3.76
5. I would highly recommend the software when no real equipment is available.	4.35
6. It would be helpful to have a version of the software running on tablet computers, such as iPad, Surface, etc.	4.18
7. I would like to have similar software for other labs.	4.35
8. I hope the software can handle more complex logic and contain more instruments.	4.06
9. I hope the software can run on Apple computers.	4.00

From Table 3, the preliminary results showed that the prototype Virtual Lab was highly effective and has great potential for other labs and courses. However, the logic engine of the prototype could only process a fixed number of three lab components at the same time. Students may not have had enough exposure to simulated laboratories because of the limited number of lab components. To solve this problem, a second version of Virtual Lab was designed and developed. Unlike the first version, the primary design goal of the second Virtual Lab was to provide a multi-component logic system. The desired functionalities of the second version Virtual Lab are illustrated in the functional diagram in Fig. 13.

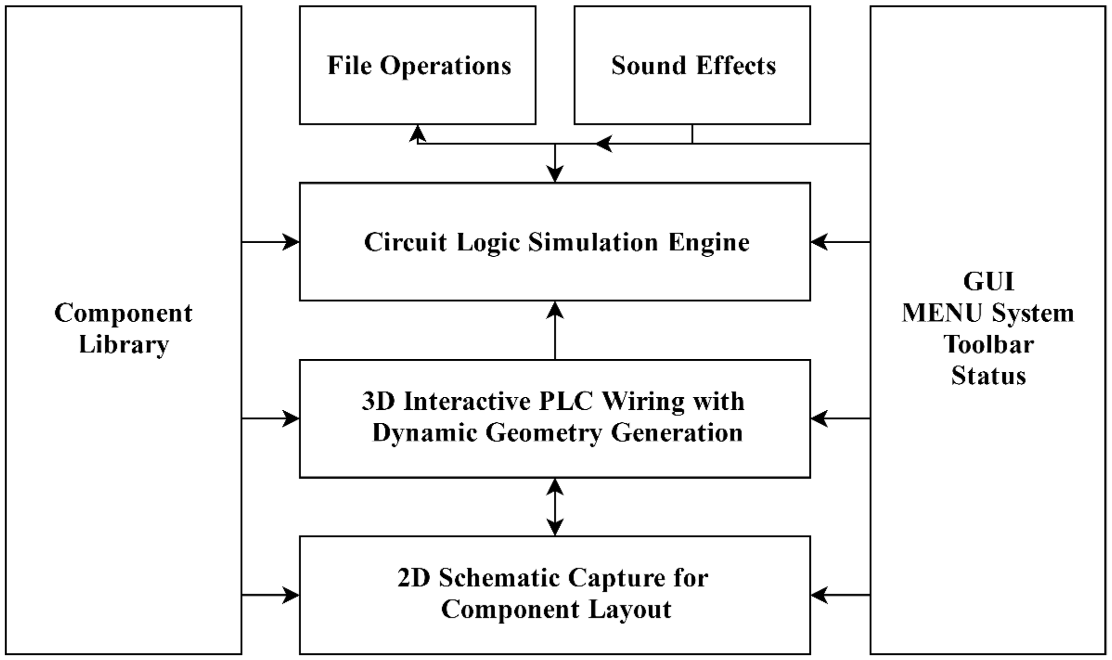

File Operations
Sound Effects
Component Library
Circuit Logic Simulation Engine
3D Interactive PLC Wiring with Dynamic Geometry Generation
2D Schematic Capture for Component Layout
GUI
MENU System
Toolbar
Status

Fig. 13. Functional Diagram of Second Version of Virtual Lab [12].

In Fig. 13, Component Library contains all the models of the components used in Virtual Lab. There are three components at the top level: Power Box, Manual Switch Box, and Electrical Relay Switch Box. Components at the top level also include small sub-components. For example, one Power Supply Box contains one switch, one LED light and two connectors. The biggest difference between the first version of Virtual Lab and the second version of Virtual Lab is that a 2D Schematic Capture for Component Layout was added to the second version to enable users to place 2D symbols of the components in the 2D schematic capture to have a clear view of the overall lab setup. Additionally, the positions of 2D symbols were used to set the positions of the 3D components.

Circuit Logic Simulation Engine is the underlying simulation engine used to simulate the circuit logic. It can automatically update all the voltages of all connected nodes (connectors) once the user hits the switch on the Power Supply Box. File Operations provide common file operations (New, Open, Save, and Save as) to the user. It has its own proprietary file format as well. Sound Effects offers a variety of aural effects for different user operations. Lastly, Graphical User Interface (GUI) provides a comprehensive graphical user interface (GUI), which contains menu systems, context menus, toolbars, and status bars. Virtual Lab also supports a large variety of user interactions such as zoom, pan, and object move.

The software architecture of the second version Virtual Lab was developed for future revisions and expansions, and it is illustrated in Fig. 14.

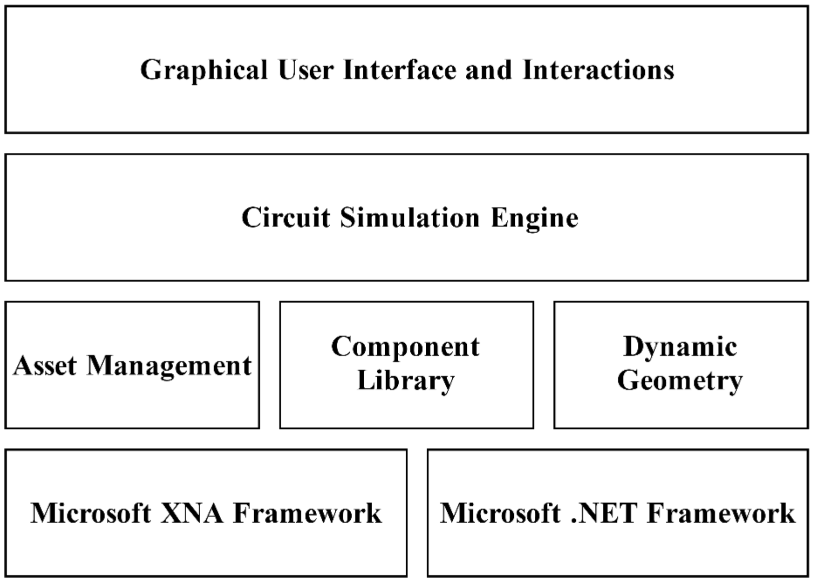


Fig. 14. Virtual Lab Version 1 and 2 Software Architecture [12].

Both Virtual Lab versions were built upon two major software libraries: Microsoft .NET Framework and Microsoft XNA Framework. The .NET framework [12] primarily runs on Microsoft Windows platforms and contains a large class library called

Framework Class Library (FCL). It also includes a common runtime engine, which is shared by all .NET-aware languages, and a base class library that provides comprehensive functionalities and services. The common runtime engine that is known as Common Language Runtime (CLR) is an application virtual machine that provides a wide range of services such as security, exception handling, and memory management. Microsoft .NET Framework is constituted by both FCL and CLR.

XNA Framework [23] is a freeware set of tools with a managed runtime environment provided by Microsoft for game development. It is based on the Microsoft .NET Framework, and can be used on various platforms, such as Windows Devices and Xbox 360. The XNA Framework manages the basics of game development and allows game developers to focus more on content and the high-level gaming experience. While game developers can use both Microsoft XNA Framework and the Microsoft .NET Framework, XNA framework is used for game-specific tasks, such as graphics rendering and managing input, and .NET framework is used for more general programming tasks.

Microsoft Windows Forms is a Graphical User Interface (GUI) library included as a part of Microsoft .NET Framework, which provides a large collection of GUI items. The GUI of the second version Virtual Lab was built upon Microsoft Windows Forms, instead of designing GUI elements from scratch. Although building custom GUIs can give an application a more unique feel and style, a small development team like the one involved in this project would be very challenged to develop custom GUIs due to the scale and complexity of the task.

Fig. 15 shows screenshots of the 2D schematic and the interactive 3D virtual wiring lab with its 3D interactive wiring environment. One advantage of second version Virtual Lab is that it increases the availability of equipment resources.

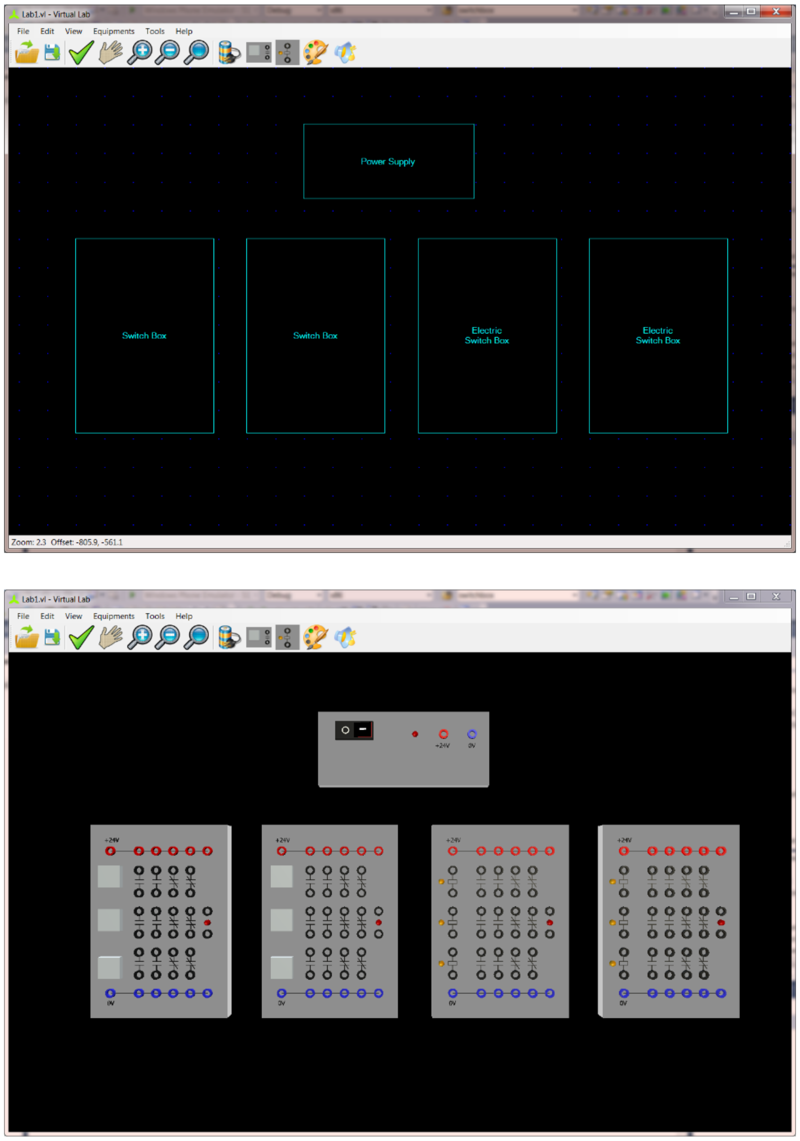


Fig. 15. Screen Capture of the 2D Schematic and 3D Interactive Wiring Environment in Second Version Virtual Lab [24].

The development of the first version of Virtual Lab started in the summer of 2013 and the first prototype was utilized in the course MET 370/386 at Old Dominion University in the fall of 2013. Since the overall feedback from students in MET 370/386 was very positive, the software was under active development based on the student feedback to include new features, such as pneumatic components.

Continuing development from the second version of Virtual Lab took a new direction, because the XNA toolset was no longer being actively developed after January 31st, 2013, and was not supported by the new "Metro interface" layers of Windows 8 or by the Windows RT platform [12]. Therefore, to continue working on the second version of Virtual Lab and to update features and functionality to meet the evolving needs of MET 370/386, the previous unfinished project was re-developed using Unity game engine in this thesis.

CHAPTER 4

METHODOLOGY

This section introduces the methodology of building Virtual Lab with a specific emphasis on the development workflow. It begins with research overview and objectives and then introduces the overall design and application architecture of Virtual Lab. Lastly, it describes the methodologies used in the application development.

4.1 Overview and Objectives

Building 3D virtual laboratories contains 5 steps: 1) observing the physical lab settings and understanding the lab procedure, (2) creating 3D models for different parts of the physical laboratory setup via 3D modeling software, (3) importing models and textures to Unity 3D and making them interactive by attaching animations and scripts, (4) implementing the software and comparing the output with real world physical equipment, (5) analyzing the results of students using the virtual lab. The overall objective of this research project is to develop a virtual lab that includes the same functionalities as the physical lab, such as making wire connections, pressing buttons, placing and rotating objects, outputting experiment results, etc. Virtual Lab should provide users with an intuitive and engaging virtual environment to study programmable logic control using mechanical, electrical, and pneumatic components.

4.2 Virtual Lab Functional Design

The primary design goal of Virtual Lab is to provide a virtual laboratory environment that is intuitive, and effective for both learning and teaching programmable logic control using mechanical, electrical, and pneumatic components. The desired functionalities of Virtual Lab are illustrated in the functional diagram shown in Fig. 16.

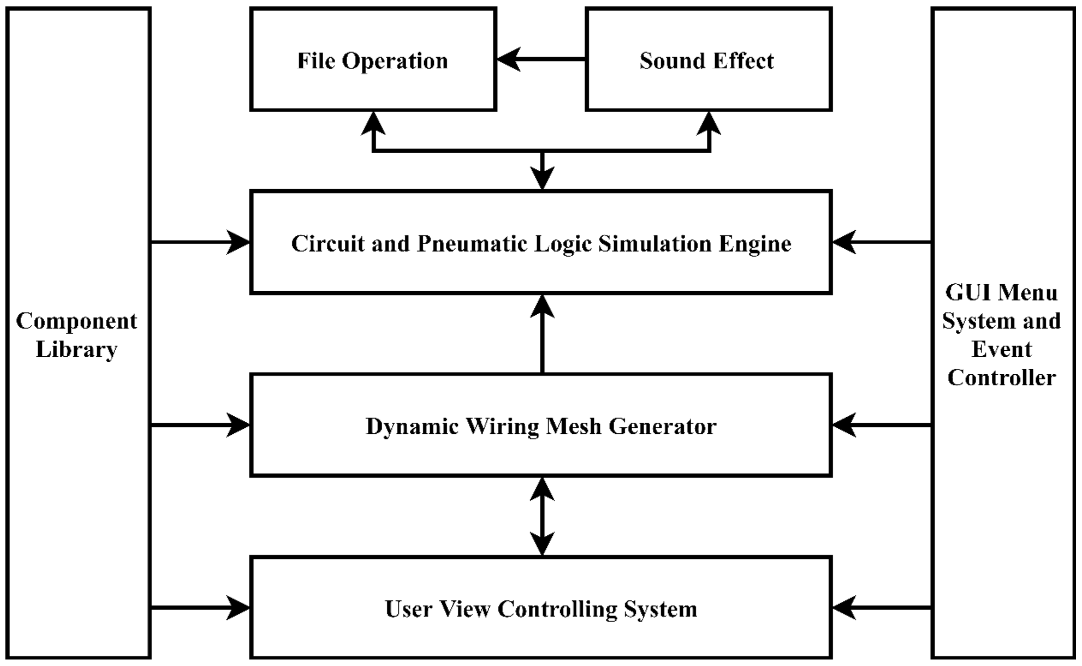

File Operation
Sound Effect
Circuit and Pneumatic Logic Simulation Engine
Component Library
GUI Menu System and Event Controller
Dynamic Wiring Mesh Generator
User View Controlling System

Fig. 16. Functional Diagram of Virtual Lab.

The details of the functional blocks are explained as follows.

- **Component Library** contains all the prefabs of the lab models that are used in the application. It contains three circuit components (power supply box, manual switch box, and electrical switch box), and nine pneumatic components (air source, single-acting cylinder, double-acting cylinder, air divider, air switch, 3/2 pneumatic valve, 5/2 solenoid control valve, 3/2 solenoid (one sided) control valve, and 5/2 solenoid (one sided) control solenoid). Moreover, each component contains other sub-components. For example, the Electric Switch Box contains electric switches, electric nodes (connectors), four LED lights, and power lines. Also, the electric switch in the Electric Switch Box contains two terminals. Finally, the application contains two types of models: one is a graphical model that gives a visual representation of the component, and the other is a behavior model that contains various behaviors of the component.

- **Circuit and Pneumatic Logic Simulation Engine** is the backbone simulation system to model and simulate the control electro-pneumatic logic. The simulation system stores and updates the connection of all the nodes (or terminals) based on the current state of the electro-pneumatic components, sets their voltage or air force correspondingly, and updates the status of the visual outputs represented by LEDs or piston rods.
- **Dynamic Wiring Mesh Generator** allows users to use the mouse to create virtual wires for connecting various electric nodes of different lab components. Although Virtual Lab has two different types of wires, electric wires and pneumatic wires, both types use same mesh constructed algorithm, and the only differences between them are size and material.
- **User View Controlling System** provides a 180° spherical view for users. It allows users to use the mouse to perform rotation, zoom, and pan actions of the camera in the application.
- **File Operation** in the application has its own proprietary file format to store lab configuration and wire connection data. Common file operations include New, Open, and Save functions.

- **Sound Effects** are also included in the application, and they are used to provide important aural cues that improve user experience and educational effectiveness.
- **GUI Menu System and Event Holder** provides a comprehensive graphical user interface that contains a menu system and toolbars. Furthermore, it monitors and manages the simulation events in the application.

4.3 Application Architecture

The application architecture for Virtual Lab was developed incorporating the desired features of Virtual Lab discussed in Section 4.2. It allows rapid application development (RAD) and facilitates future revisions and expansions. The application was developed using object-oriented programming with C# programming language, and the current implementation contains more than 30 classes and types. The application architecture is shown in Fig. 17.

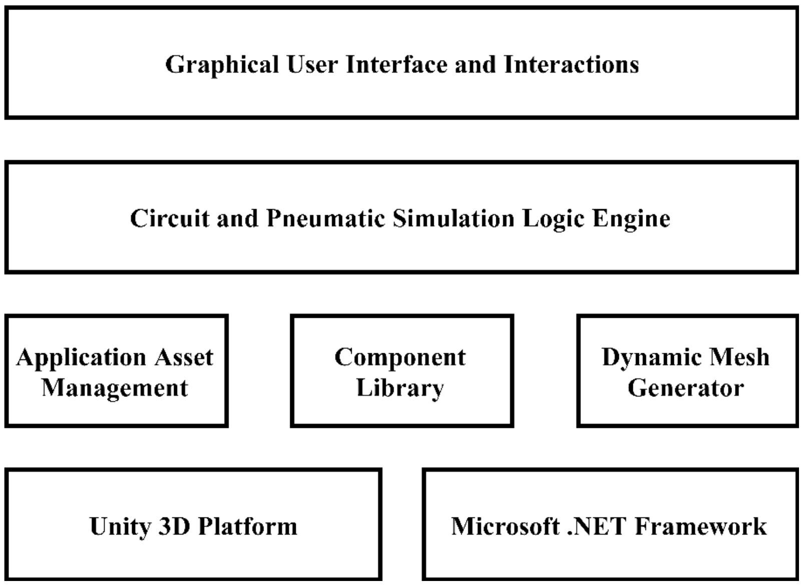

Graphical User Interface and Interactions
Circuit and Pneumatic Simulation Logic Engine
Application Asset Management
Component Library
Dynamic Mesh Generator
Unity 3D Platform
Microsoft .NET Framework

Fig. 17. Virtual Lab Application Architecture.

The major application components utilized by or developed for Virtual Lab are shown in Fig. 17. The details of Microsoft .NET Frame will not be presented here because it has been discussed on Page 26, Section 3. The details of the remaining components are described as follows.

- **Unity 3D Platform** is a cross-platform game engine developed by Unity Technologies. Since Unity was announced at Apple's Worldwide Developers Conference in 2005, it has been extended to 21 hardware platforms. With an emphasis on portability, it targets a large amount of application program interfaces (APIs), such as Direct3D on Windows and Xbox 360; OpenGL on Mac, Linux, and Windows; OpenGL ES on Android and iOS; and proprietary APIs on video game consoles. The advantages of using Unity 3D include providing specification of texture compression and resolution settings for various game-engine-supported platforms and providing support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects. Additionally, the graphics engine's platform diversity within Unity 3D provides a shader with multiple variants and a declarative fallback specification, allowing Unity to detect the best variant for the current video hardware and, if none are compatible, to revert to an alternative shader that may sacrifice features for performance. Also, in a Unity project, developers have control over delivery to various platforms, such as PCs, consoles, mobile devices and websites [24].

- **Asset Management** contains and manages all the digital assets used in Virtual Lab, including raw 3D models, 2D image textures, materials, modified prefabs, and sound effects. Unity 3D supports importing models from most popular 3D applications. Most of the 3D models are pre-processed into modified prefabs in order to be adapted to Virtual Lab scenario and facilitate user operations. Moreover, this thesis developed a custom mesh generator in Virtual Lab for generating dynamic cube meshes.
- **Component Library** contains both graphical models and behavior models for the components. Models are represented in various dimensions: 2D symbols in the main menu are shown for equipment, while connectors (or terminals), switches, electric and pneumatic wires, circuit equipment and pneumatic equipment are shown in the 3D virtual environment. Many C# classes were utilized to represent and manage different components. Furthermore, software reuse techniques such as inheritance and containment have been heavily utilized in building the component library.

- **Dynamic Mesh Generator** contains a vital component in Virtual Lab, which creates a dynamic tube-shaped mesh for connecting different connectors on the equipment in Virtual Lab. There are two types of geometrical objects used to display the wires. After the user selects the two connectors (nodes) from the equipment, an adjustable curve is formed and rendered by the movement of the cursor. Different than the previous versions of Virtual Lab, this adjustable curve is a simple polyline generated using Bézier curves instead of Hermite curves. If the user clicks the mouse left button, the wire location is finalized and becomes a thick wire object. The thick wire object is represented by a triangle mesh that is generated by extruding the Bézier curve from the center. Additionally, the current dynamic geometry generation for mesh generator is implemented on Central Processing Units (CPU).
- **Circuit and Pneumatic Simulation Logic Engine** provides the core simulation logic for the programmable logic controller in Virtual Lab. A connector (node) is the most fundamental component in the circuit and pneumatic simulation logic engine. Each node includes three states based on its current voltage: Positive, Ground, and None. The topology of the electro-pneumatic connection is determined and updated based on the generated wires and the status of the switches. The nodes states are then updated according to the current electro-pneumatic topology. The final simulation outputs in the form of LED lights, and pistons are updated based on the states of nodes.
- **Graphical User Interface and Interactions** contains a Unity 3D User Interface (UI) [25] system which allows developers to create user interface quickly and

intuitively. It contains various UI elements including: UI button, UI Canvas, UI image, etc. The most important part in UI system is Canvas, which is the area that includes all the UI elements. The Canvas is a Game Object with a Canvas component on it, and all UI elements are children of this Canvas. Canvas has three render mode settings: Screen Space-Overlay, Screen Space-Camera, and World Space. The render mode is used to determine if the user interface will be rendered in screen space or world space. In addition, an event system will also be attached to the Hierarchy once a Canvas is setup. The primary functionality of the event system is to send events to objects in the application based on user inputs, such as keyboard, mouse, touch, and custom input. The Event System consists of a few components that work in conjunction to send events.

CHAPTER 5

IMPLEMENTATION

This section describes the implementation of Virtual Lab. It begins with the selection of the development platform used for Virtual Lab, followed by 3D modeling of the components in Virtual Lab. It then presents the design of the simulation logic engine. Lastly, it describes the methodologies used in virtual wires generation.

5.1 Development Platform

When choosing the development platform for Virtual Lab, the primary factor for not choosing 2D graphics tools is that they cannot provide users a realistic three-dimensional environment and equipment. Autodesk MAYA and Autodesk CAD are two popular computer programs for 3D modeling, and they are used to create, animate and implement 3D models in immersive environments. Autodesk MAYA is a powerful software tool widely used by top feature film animation studios and game development companies. It offers 3D modeling, animation, simulation, and rendering tools on an extendable production platform. CAD, also called Computer-Aided Design, utilizes computer technology for design and design documentation. Autodesk CAD was the first CAD program in history. It is extensively used in architecture and structural engineering fields. It allows engineers to visualize their designed concepts through visual renderings and to simulate how their designed models will perform in the physical world.

Unity is the one of the most widely used game engines in the video game industry. It provides many features and an intuitive and extensive interface. Unity supports assets from major 3D modeling tools, and all the 3D models raw files from

Autodesk MAYA, Blender, and 3D MAX can be directly imported to Unity [26]. Moreover, most of Unity's API and project structure is identical for all supported platforms, and in some cases a project can simply be rebuilt to run on different devices.

As previously mentioned, 3D models generated by Autodesk are used to create a virtual lab environment in Unity. Script languages must be utilized to interact with 3D models, and these scripts can be used to store and manage the various types of data within the models. Developers can access different parts of 3D models and easily control them via scripts. When developing M&S applications in Unity, scenes are containers for all types of objects that are essential to the application. In each scene, programmers can design and construct their specific level/section of the application separately. Virtual Lab contains three scenes in total: Start Scene (Menu), Main Scene (Equipment Operation) and Side Scene (Equipment Inspection). Start Scene acts as a portal and presents users with various menu options. Main Scene allows users to interact with different circuit and pneumatic equipment models and to perform virtual experiments. Additional technical information about each piece of equipment in Virtual Lab is included in the Side Scene.

5.2 3D Modeling of Virtual Lab Components

In computer graphics, 3D modeling is the process of developing a mathematical representation of a three-dimensional surface of an object through specialized software. 3D models can be created by connecting a collection of points in 3D space via various geometric structures such as triangles, lines, curved surfaces, etc. The two main approaches to 3D modeling are Polygon modeling and NURBS. Polygonal modeling uses polygons to represent the surfaces of 3D models. A group of polygons that are connected by shared vertices is referred to as a mesh. A triangular mesh is a surface object consisting of arranged triangles, each of which is defined by three vertices. Multiple triangles can generate more complex polygons, and a group of polygons is referred to as an element. A single polygon within the element is called a face. To display a mesh in three-dimensional space, the mesh cannot pierce itself. For instance, an edge, which is two vertices connected by a straight line, cannot pass through a polygon [26]. The main advantage of polygonal modeling method is that the representation speed is faster compared with other modeling approaches. However, polygonal modeling is incapable of accurately representing curves and curved surfaces.

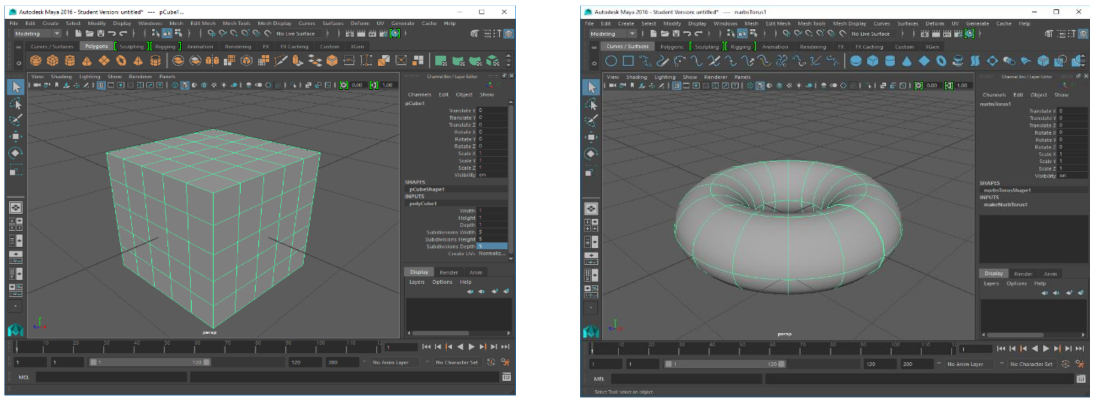


Fig. 18. Polygon & NURBS Modeling in MAYA.

Non-uniform Rational Basis Spline (NURBS) [27] is another 3D modeling technique commonly used in computer graphics. NURBS modeling provides smoother and more mathematically precise curves and curved surfaces. Thus, it is mainly used in automotive design and industrial parts design. Fig. 18 shows the polygon and NURBS modeling in MAYA. The key benefit of using NURBS is that it uses a more precise mathematical calculation technique to smoothly and accurately display a curved surface.

Most pneumatic equipment is created in Autodesk CAD using NURBS to keep the original data exact and complete, without any loss of definition. However, Unity does not directly support the following 3D modeling packages: NURBS, Non-Uniform Rational Mesh Smooth (NURMS), and subdivision surfaces. Therefore, these models must be converted into polygons before importation to Unity. Autodesk MAYA has the capability to convert a NURBS surface to polygon mesh, and the process of converting NURBS to polygons is to select the NURBS surface from the 3D model and choose Modify > Convert > NURBS to Polygons [28].

In some special situations, converting a NURBS surface to a polygon mesh can result in some polygon normals of the converted mesh facing wrong directions. To solve this problem in Autodesk MAYA, Reverse function was used to reverse the face normal on a polygon face and switched the “front” and “back” facing direction of the polygon normals. The green lines in the following Fig. 19 shows the directions of polygon normals that will be rendered in the 3D Space.

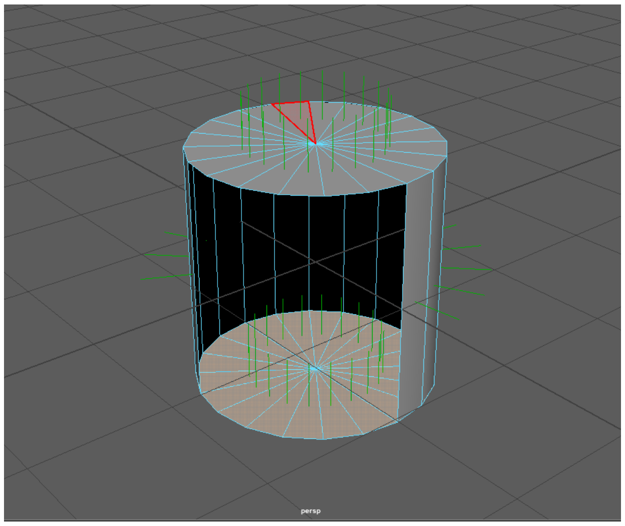
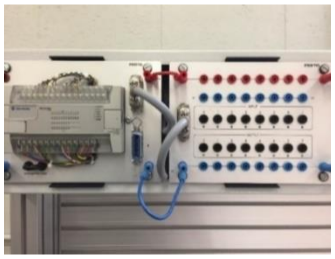
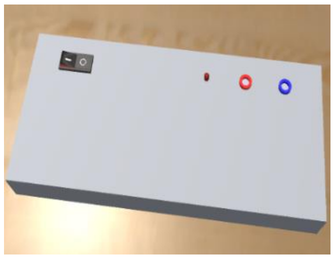
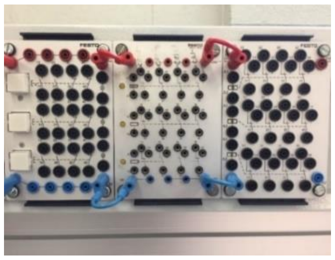
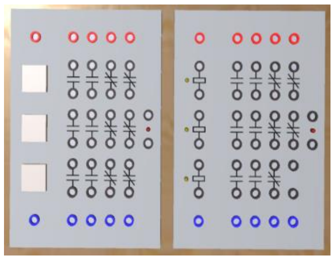
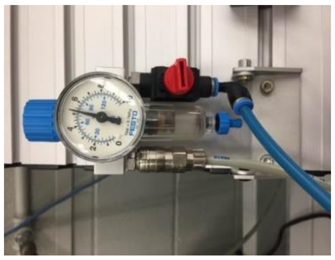
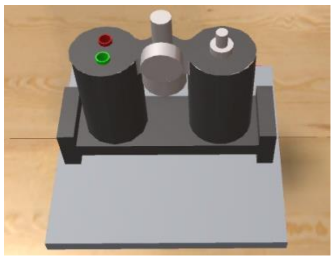
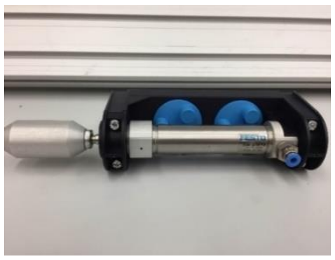
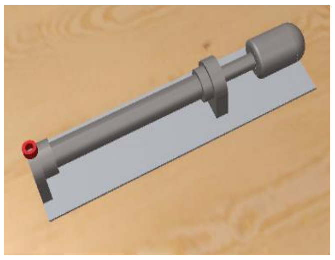


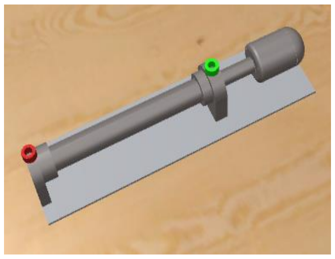
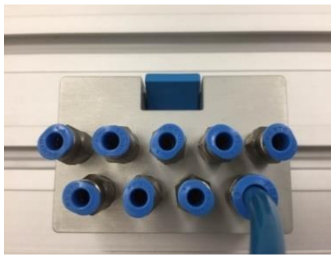
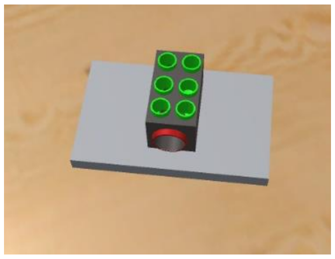
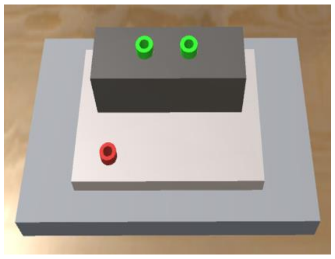
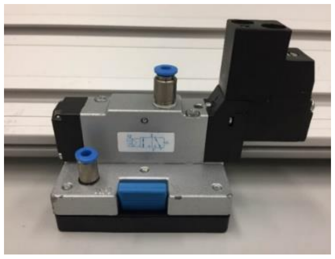
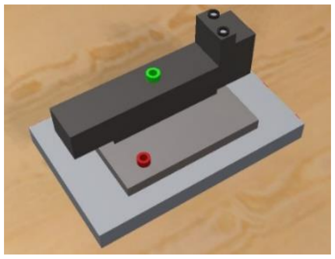
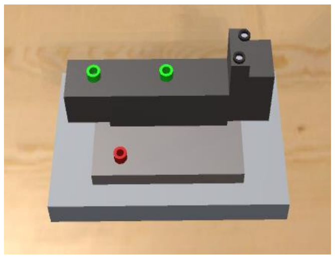
Fig. 19. Reverse Polygon Normals.

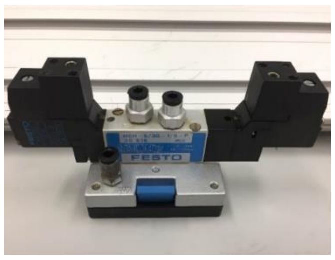
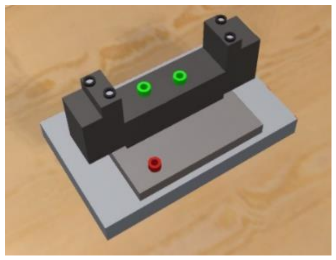
Table 4 shows the physical equipment in the lab and the corresponding 3D models generated in Maya.

TABLE 4

TABLE OF 3D MODELS AND PHYSICAL EQUIPMENT COMPARISON

Physical Equipment	3D Virtual Model	Equipment Name
		Power Supply Box
		Manual Switch Box and Electrical Relay Switch Box
		Air Supply
		Single-Acting Cylinder

		Double-Acting Cylinder
		Air Divider
Physical Component Not Available		3/2 Directional Control Valve
		3/2 Solenoid (One Sided) Control Valve
Physical Component Not Available		5/2 Solenoid (One Sided) Control Solenoid Valve

		5/2 Solenoid Control Valve

5.3 Simulation Logic Engine Design

Since there is a large number of components in Virtual Lab, creating a Simulation Logic Engine (SLE) to hold all the application objects is essential. The purpose of creating a SLE is fourfold. The first goal is to store and manage all types of data for the various virtual objects. The second goal is to generate realistic virtual simulation logic for the application. The third goal is to monitor various simulation events and modify them if necessary. The fourth goal is to track and edit specific data as required.

There are two types of simulation in Virtual Lab: electric and pneumatic control simulation. In the physical world, an electric circuit [29] is the combination of individual electric components, such as resistors, switches, inductors, transistors, and capacitors, which are connected by conductive wires that allow an electric current to flow. This combination of components and wires can perform various operations, like amplifying signals, transferring data, and performing computations. Compared with the physical experiment, the functionality of each equipment in Virtual Lab is the same. Various data structures are used to store various data to create realistic physical reactions, and algorithms are utilized to perform calculations, data processing, and automated actions.

In an electromagnetic field [30], an electric charge is the basic property of elementary particles of matter, which causes all electric and magnetic forces and interactions. Electric charges may be positive or negative, and an electric current occurs when positive and negative charges are placed on the opposite ends of an object. Every electric component should be given a customized data structure that includes its current voltage and other essential information as a property, and data of each electric component are used to determine if any simulated reaction should occur. For example, a LED light contains two electrical parts A and B, and both A and B are given a property to store their current voltage, etc. An illustration of this is shown in Fig. 20.

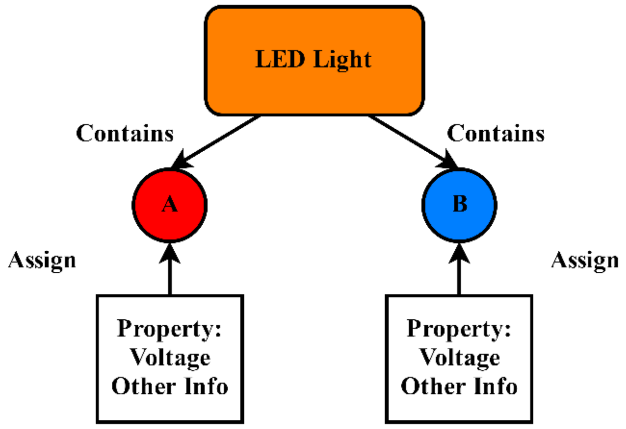


Fig. 20. Illustration for Circuit Design.

To determine if an LED is activated, the combination of A and B must include one positive and one negative charge. If this condition is satisfied, then the electric current may virtually pass through the LED. The pseudocode for the LED is shown in Fig. 21:

```
function LED Switch (Time timer)
      A.Voltage ← Voltage.None
      B.Voltage ← Voltage.None
      if Circuit is turned on
            UpdateVoltage(A and B)
            if A.Voltage == Voltage.Possitive && B.Voltage ==
            Voltage.Negative then
                  LED.Switch ← On
            else if A.Voltage == Voltage.Negative && B.Voltage ==
            Voltage.Positive then
                  LED.Switch ← On
            else
                  LED.Switch ← Off
            end if
      end if
```

Fig. 21. Pseudocode for the LED.

In addition, users may encounter short circuits during the experiment. A short circuit [31] is an electrical circuit that allows a current to go through an unintended path with almost no electrical resistance. This results in an excessive electric current and potentially causes circuit damage, overheating, or even fire and explosion. When a short circuit occurs, two nodes of this electrical circuit must be abnormally connected, and the voltages of each intended to be different.

As previously mentioned, every electric component is given a customized data structure that includes its current voltage and other essential information as a property. A short circuit will happen when a connection between two nodes forces them to have the same voltage. An illustration of this is shown in Fig. 22 as follows:

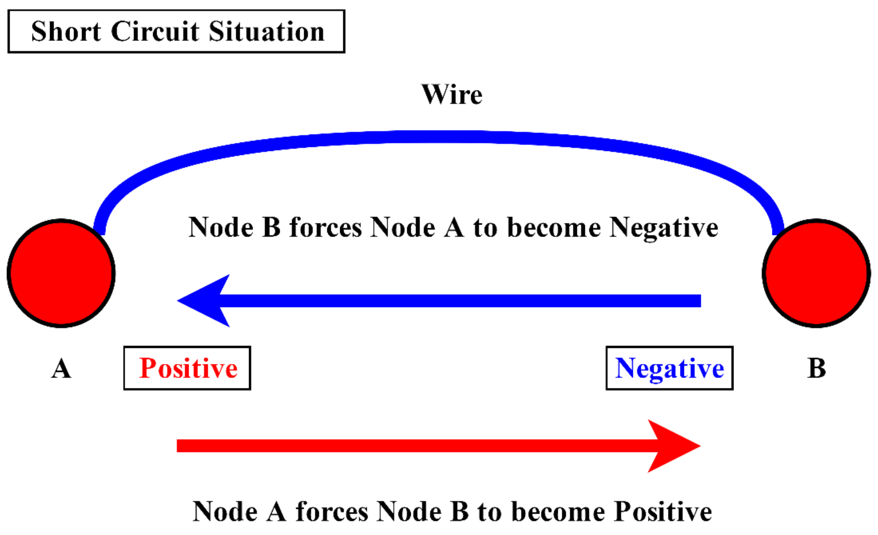

Short Circuit Situation
Wire
Node B forces Node A to become Negative
A
Positive
Negative
B
Node A forces Node B to become Positive

Fig. 22. Illustration of a Short Circuit.

From the figure above, node A is positive and node B is negative, and both force each other to change their voltages to the same value. If this condition is satisfied, then a short circuit occurs.

Pneumatic systems are used extensively in industrial areas and are commonly powered by compressed air or other gases [32]. A pneumatic system can be controlled through selected solenoid valves, and it provides a lower cost, more flexible, safer alternative to electric motors and actuators. With pneumatics, air is usually pumped into a receiver using a compressor and then the receiver holds a large volume of compressed air to be used by the pneumatic system as needed. Pneumatic systems also use a variety of valves for controlling pressure, direction, and speed of actuators.

As mentioned in Section 4.3.10, every electric component is given a customized data structure that includes its current voltage and other essential information as a property. It is essential to apply the same data structure method to every pneumatic unit. However, unlike the circuit simulation, there is no electric charge in any pneumatic unit.

Therefore, to simulate the air flow passing through various pneumatic components in Virtual Lab, the property of voltage in each pneumatic unit was replaced with the property of air pressure. Compared with the property of voltage which has three states, the property of air pressure has only two states: positive and none. For example, a double-acting cylinder contains two air inlets A and B, and both A and B are given a property to store their air pressure states, etc. An illustration of this is shown in Fig. 23.

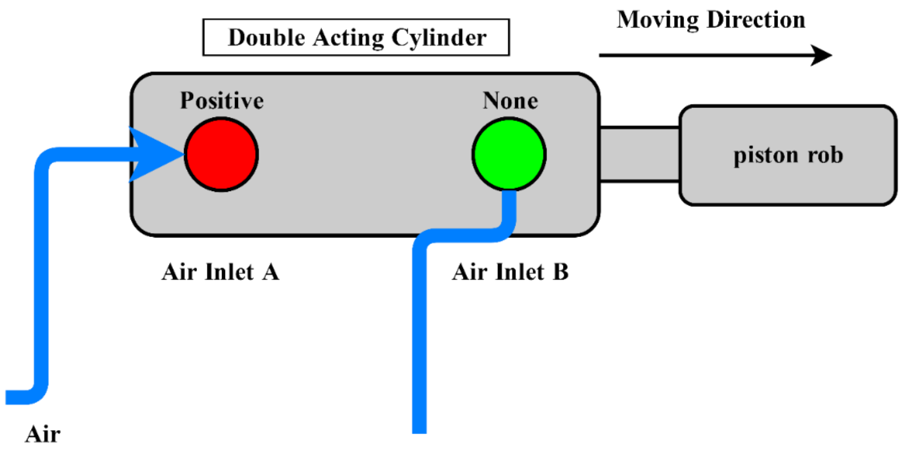


Fig. 23. Air Force States in Double-acting Cylinder.

The moving direction of a piston rod is determined by the status of both air inlet A and B. When air goes into air inlet A, the status of air inlet A will be become positive. The status of air inlet B remains none if there is no air force going into B. The logic of moving direction is shown below:

$$MovingDirection = \begin{cases} moving\ left\ (A.state = positive\ and\ B.state = none) \\ no\ movement \qquad (A.state = B.state) \\ moving\ right\ (B.state = positive\ and\ A.state = none) \end{cases} . \qquad (1)$$

The electro-pneumatic system is controlled by an electric current and operated by opening and closing valves. In many electro-pneumatic systems, the devices being controlled are electrically-actuated directional control valves. These control valves supply air pressure to devices such as double-acting cylinders that will extend or retract a rod when pressure is applied or removed. Built-in solenoids are used to open and close these valves and are activated with AC or DC voltage signals. For example, a solenoid contains two air outlets A and B, and an air inlet C. A will be connected to C if left valve is on, and B will be connected to C if right valve is on. However, air can only go from air inlet C to air outlets A and B, and the process cannot be reversed. The illustration is shown in Fig. 24.

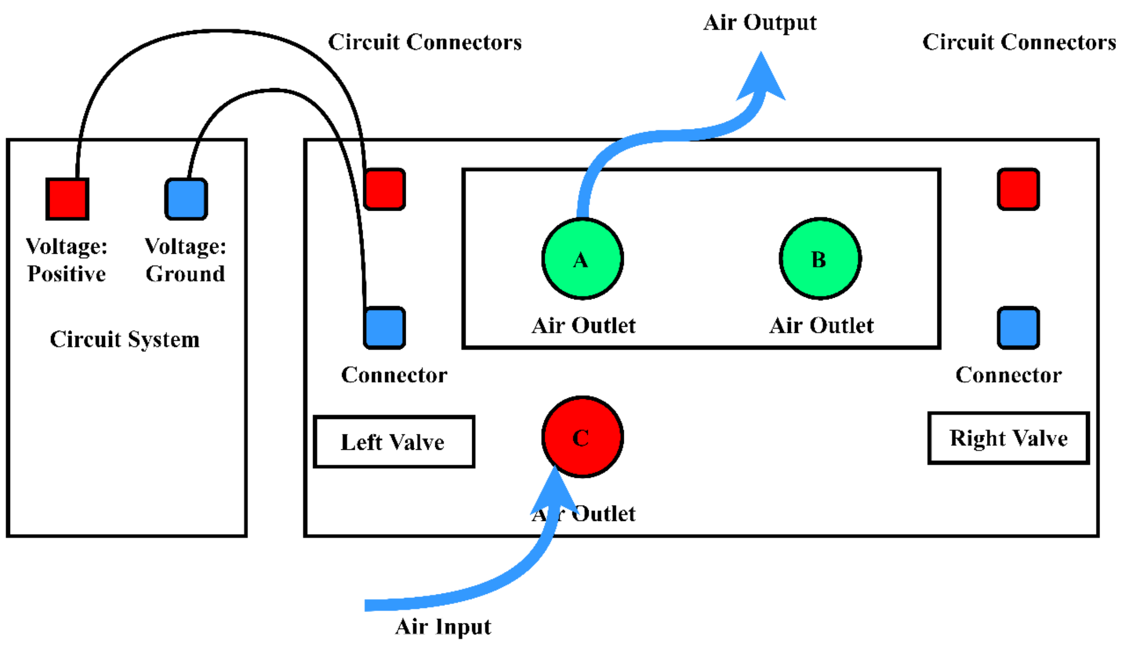


Fig. 24. The Functionality of a Solenoid.

The left valve is on when the combination of connectors in the left side contain one positive and one negative charge. In that case, air inlet C is connected to air outlet A, and air can go through C to A.

After students connect various components in Virtual Lab, this application can simulate the PLCs accurately. The fundamental data structure behind this simulation is a graph data structure. In mathematics, a graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges. Furthermore, in computer science, a mathematical graph is referred to as Graph Data Structure [33], and it can be represented by two elements: an array of vertices and an array of edges. Each node of the graph is represented as a vertex. Edge represents a path between two vertices or a line between two vertices. There are different types of edges in Graph Data Structure. One is directed versus undirected edges, and another one is weighted versus unweighted edges. By default, an edge represents the connection between one node and another, and this connection is assumed to be bidirectional. That is, if there exists an edge between nodes A and B, it is assumed that one can travel from A to B and from B to A. Graphs with bidirectional edges are said to be undirected graphs, because there is no implicit direction in their edges. In some cases, though, an edge might infer a one-way connection from one node to another. For example, the earth is illuminated by the sun light, the travel path of the sun light would imply that the edge between the earth and the sun would be unidirectional. That is, graphs with unidirectional edges are said to be directed graphs. Furthermore, because graphs represent a collection of objects and their relationship among these objects, it can be important to assign a value between the connection from one node to another, in the form of graphs with weighted edges. For example, if one considers the map of the United States, the value of any path from one city to another is the sum of the value of the edges along the path. The shortest path, then, would be the path with the least value. Lastly, two vertices are termed as adjacent if they are connected to each other through an edge. For example, B is adjacent to A, and E is adjacent to A, etc.

The same method has been utilized in circuit simulation and pneumatic simulation to save and update all the nodes (connectors) that users have connected. Once all the nodes have been selected, the information of each node and its adjacent nodes is saved. In circuit simulation, when users press the power button in the simulation environment, the update function starts to run. The first update function is to first check the number of equipment that are inside the virtual circuit. Then it will locate all the power nodes that are connected in the power box and update the red nodes' current voltages to positive and the blue nodes' current voltages to ground. The second step is to update all the adjacent nodes of the power nodes. Each node stores a list of its neighbor nodes so that after all the power nodes are updated, the first neighbor node of each power node will be updated as well. The update function in each node contains a recursive implementation of a Depth-first search (DFS) algorithm which causes a selecting node to start updating itself first and then explore its neighbor nodes as far as possible along each branch before backtracking to the previous node. The example of the updating process is shown in Fig. 25:

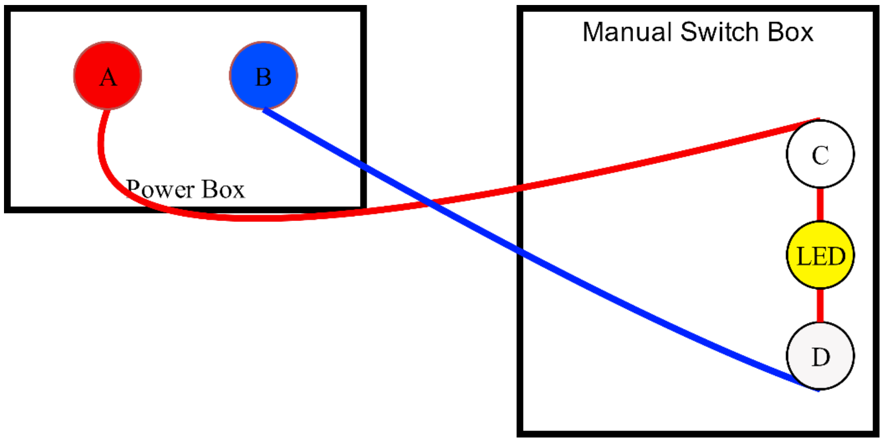


Fig. 25. The Example of Update Process.

From Fig. 25, it shows that node A of the power box connects with node C of the manual switch box, node C and node D are the two sides of the LED light. Node B connects with node D. After the user presses the power button, the current voltage of node A will be updated to positive and the current voltage of node B will become ground. Because node A and node B are also connecting with other neighbor nodes, the voltages of their neighbor nodes will be updated as well. In this example, node C connects with node A, so its voltage will be updated to positive. The same case applies to node D, its voltage will be changed to ground. After finishing updating the circuit logic, the LED will be activated due to its two nodes including one positive and one negative charge. The same principle is being applied to the pneumatic simulation, but the only difference between these two simulations is that the voltage states of the pneumatic node does not contain ground. In addition, there is no short circuit scenario in the pneumatic system.

5.4 Virtual Wire Generation

In the physical world, wires [41] are flexible cords commonly used to connect various equipment and carry electricity or telecommunication signals. In Virtual Lab, wires act as conduits for data transmission and represent a link between two virtual connectors of the lab equipment. Virtual Lab enables users to draw customized wires in the simulated environment by configuring the coordinates of the two selected virtual connectors and the current mouse position.

Before constructing virtual wires in Virtual Lab, a Mesh class is created to generate customized mesh. Within the Mesh class, a single array is used to store all the vertices of a mesh, and each triangle is specified by using three integers that correspond to indices of the vertex array. The triangles are also put together into a single array of integers, and each of them is specified as triples of integers in that array. The first three elements in the array define the first triangle, the next three define the second triangle, and so on. Moreover, the sequence of the corner vertices of each triangle should be arranged in a clockwise sequence so that the visible outer surface is correctly displayed. In addition, adding normals to a mesh that is generated through the Mesh class can aid in correctly shading the mesh. The illustration of mesh generation workflow is shown in Fig. 26 below:

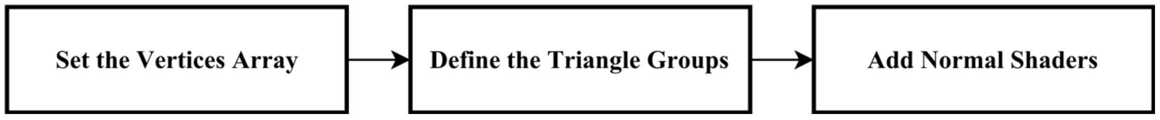


Fig. 26. Mesh Generation Workflow.

Creating a customized tube-shaped mesh object in Unity 3D engine requires all the vertices of that object in the 3D space. After the application system obtains the essential information, a Bezier curve mesh will be displayed at runtime. A Bezier curve is a parametric curve frequently used in computer graphics and related fields, and it is defined by a group of control points through P_0 to P_n, where n is the order number and P_0 and P_n are the start point and end point of the curve. Thus, the intermediate control points that determine the curvature of the wire generally do not lie on the curve itself. Users can adjust the curvature of the Bezier curve by changing the position of its intermediate points. The illustration is shown in Fig. 27 as follows:

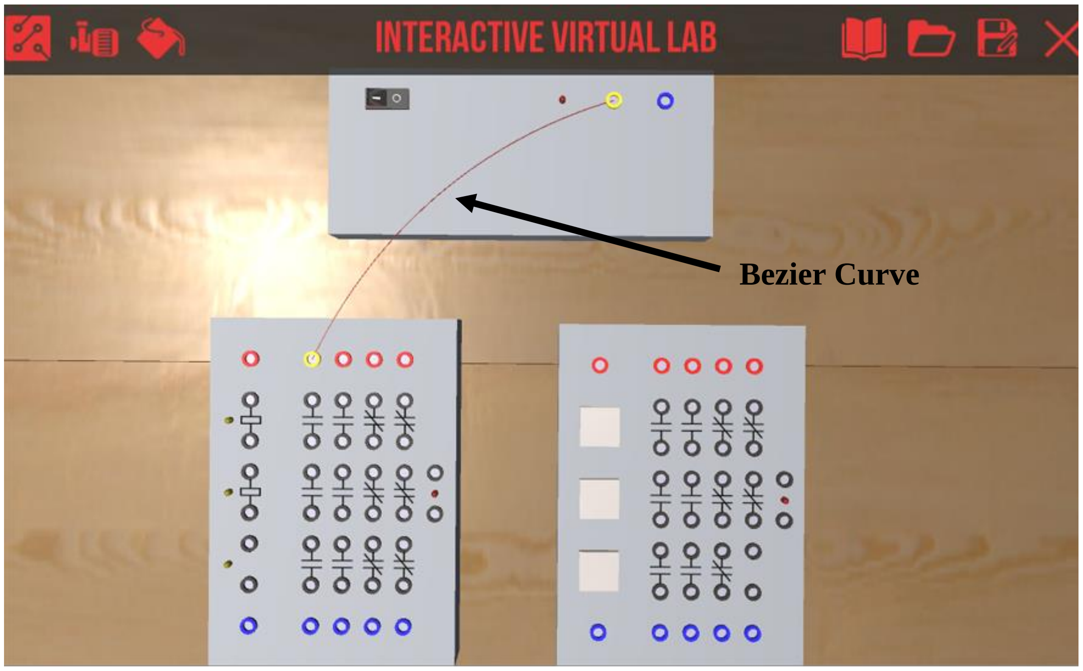


Fig. 27. Bezier Curve in the Runtime Application.

In order to create a high-quality Bezier curve, a type of Bezier curve called a quadratic Bezier curve [34, 35] can be utilized to calculate all vertices on the curve. A quadratic Bezier curve is created by a function B(t), which contains three given points: P_0, P_1, and P_2. A quadratic Bezier curve can be divided into a large number of segments to generate a life-like, high quality wire rendering,

$$B(t) = (1-t)[(1-t)P_0 + tP_1] + t[P_1(1-t) + tP_2], 0 \le t \le 1. \qquad (2)$$

Based on the equation above, P_0 and P_2 are the coordinates of the end points, and P_1 is the coordinate of intermediate control point. As t increases from 0 to 1, the curve departs from P_0 in the direction of P_1. Parameter t from the function B(t) can be used to determine the coordinates of all the segment points. For instance, assuming a Bezier curve is separated into n segments, and the coordinates of P_0, P_1 and P_2 are gained from the application system, the coordinate of P_i can be calculated by the equation below:

$$P_i = \left(1 - \frac{i-1}{n}\right)\left[\left(1 - \frac{i-1}{n}\right)P_0 + \frac{i-1}{n}P_1\right] + \frac{i-1}{n}[P_1\left(1 - \frac{i-1}{n}\right) + \frac{i-1}{n}P_2], 0 \le \frac{i-1}{n} \le 1. \qquad (3)$$

To compute the vertices around P_i, the vectors between three closed points must be calculated. As indicated by Fig. 28 below, V_1 can be calculated from P_{i-1} and P_i, while V_2 can be calculated by P_i and P_{i+1}. T_1 will be computed through V_1 and V_2. V_3 is the cross product of V_1 and V_2. To increase the accuracy in determining side vertices' coordinates, a new V_4, which is perpendicular to V_3 and T_1, was also calculated as a cross product of V_3 and T_1. With the given calculated values from V_3 and V_4, the positions of side vertices that circle around P_i can now be located.

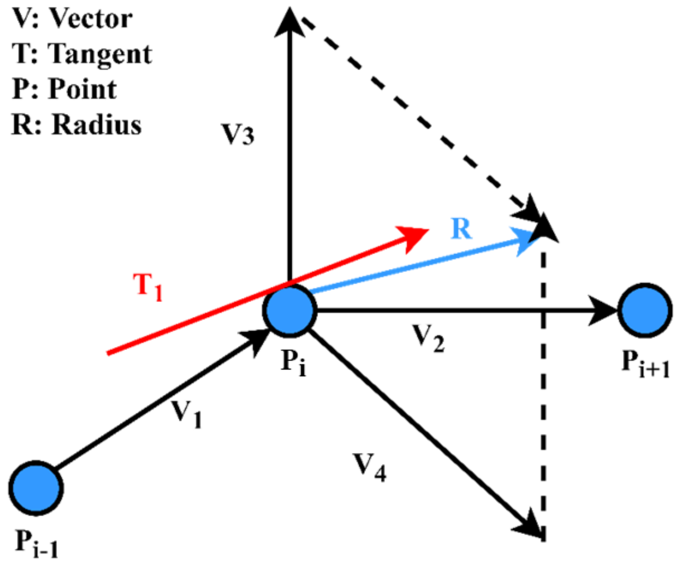


Fig. 28. Coordinate for Side Vertex.

V_3 and V_4 must be normalized before calculating the target vectors between P_i and the side vertices of P_i that are based on two given scalar values: R and α. R defines the radius of the circle around the P_i while α indicates the angle between V_3 and the target vector. For instance, constructing a virtual circle in 3D space around a point P_i requires eight side vertices (named from SV_0 to SV_7). Similarly, the target vector between the target vertex and P_i is named from TV_0 to TV_7. The instructions are shown in Fig. 29.

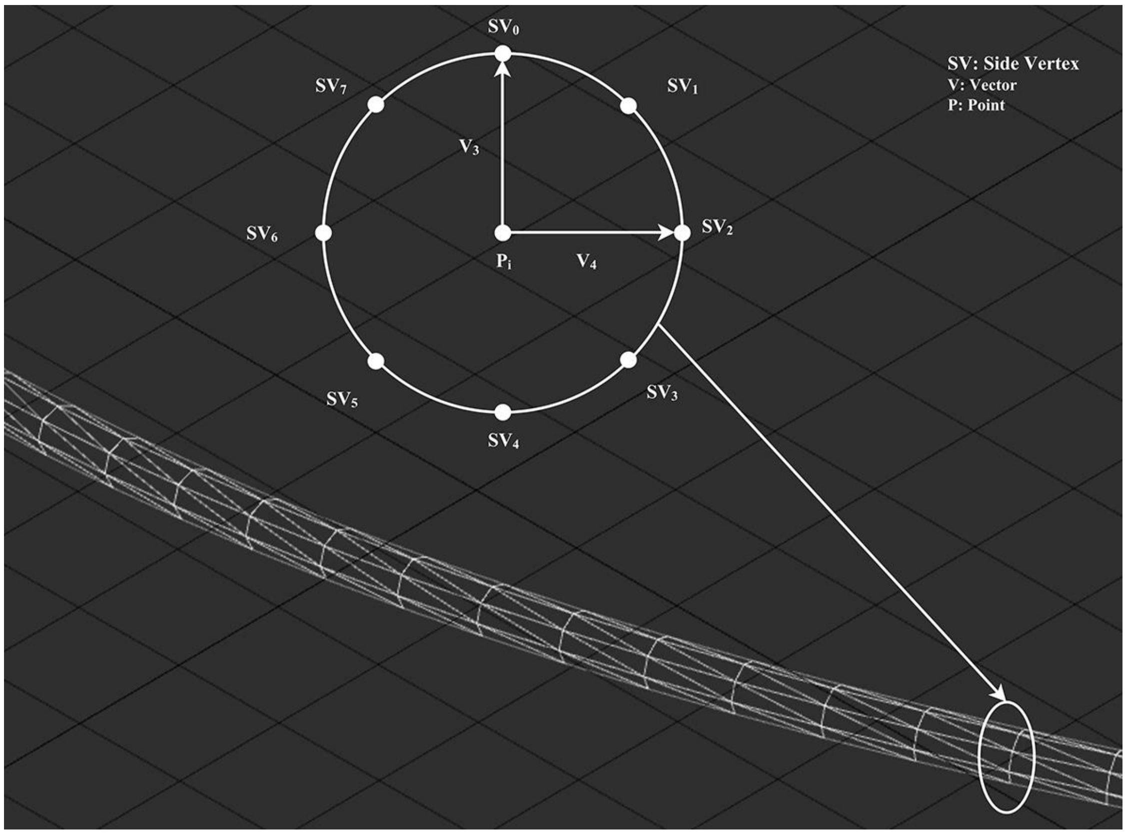


Fig. 29. Target Vectors Calculation.

Once the system analyzes the results of all the target vectors, it will perform the final calculation (shown in Equation 4) for the actual coordinates of all the side vertices around P_i (illustrated in Fig. 30).

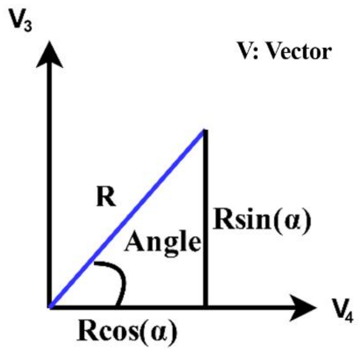


Fig. 30. Side Vertices Calculation.

$$SV_a = P_i + R * sin(\propto) * V_3 + R * cos(\propto) * V_4 \qquad (4)$$
$$0 \leq a \leq 7$$

Repeat the procedure above to generate a tube-shaped mesh, and the result of the wire object in Virtual Lab is shown in Fig. 31.

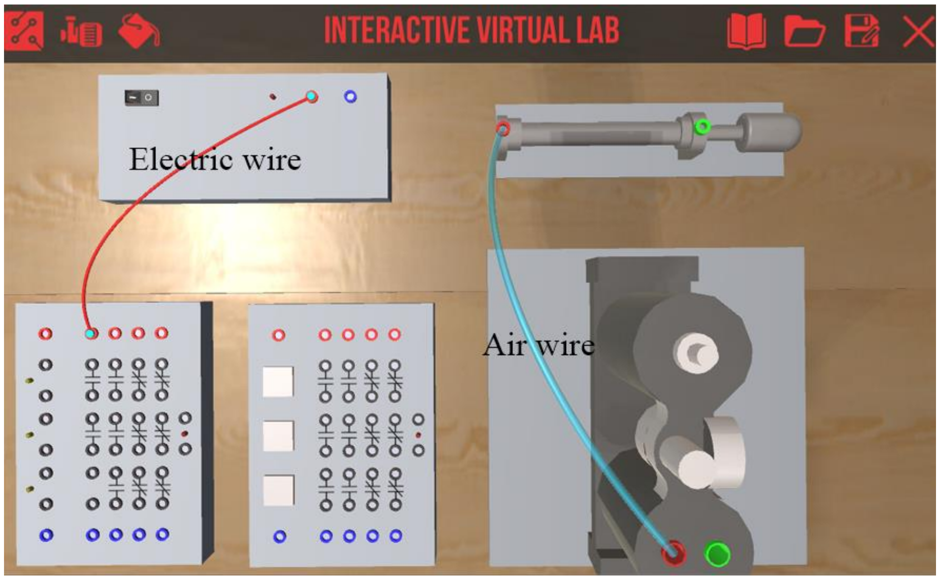


Fig. 31. Generated Wires in Unity 3D.

5.5 User Interactions

In the real lab, students are able to position and rotate equipment at desired locations. Therefore, a User View Controlling System is a new feature in Virtual Lab. It provides a 180° view to users, which they can manipulate and customize using the mouse. The user can use the mouse right click to perform rotation action and the mouse middle button to perform pan action. In addition, the mouse scroll wheel is used to perform zoom action. Since computer screen space is defined in pixels, the bottom-left of the screen indicates (0, 0), while the right-top of the screen is (SceenPixelWidth, SceenPixelHeight). The third parameter z is the distance from the camera. However, the real coordinate for positioning the object in a 3D world is quite different from the camera view. The illustration is shown in Fig. 32. Unity [26] provides a function called ScreenToWorldPoint, which can transfer the mouse position from the screen space into the world space.

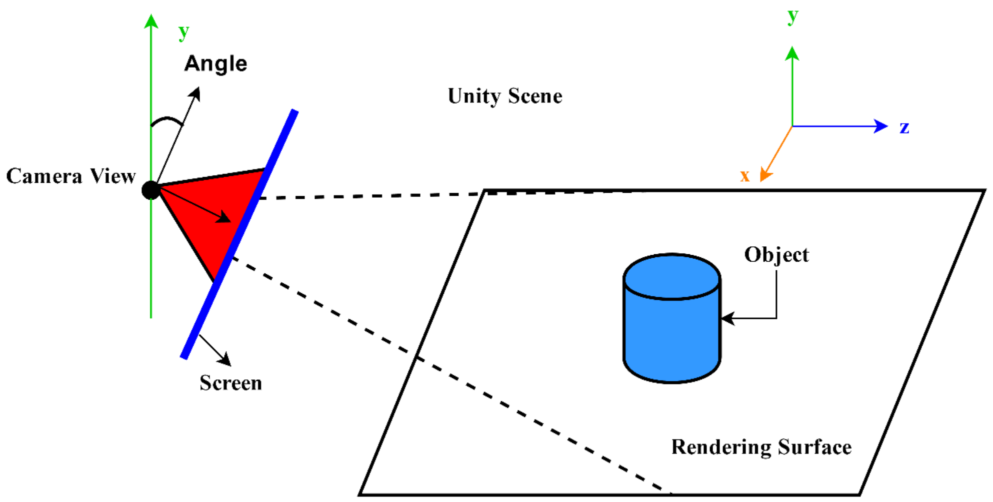


Fig. 32. Screen Space and 3D World Space.

To determine if a specific virtual object is clicked by a user, there are two methods to highlight a selected object. The first method is to modify the color value from the material of the selected object (Fig. 33). The second method is to create and apply a special shader called Silhouette-Outlined [36] to the material of the selected object.

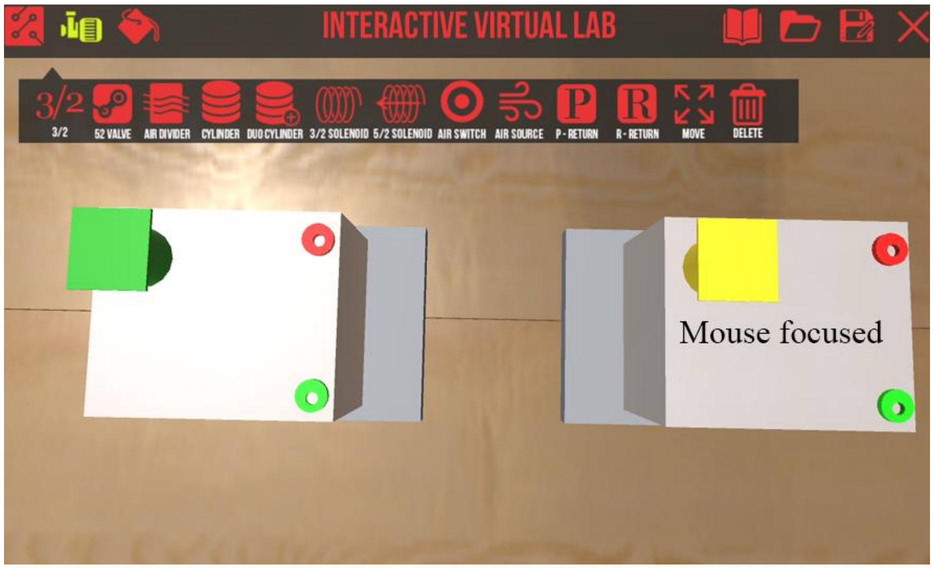


Fig. 33. Color Value Modification.

Modifying color value from the material to highlight the selected object is efficient when the object does not have multiple materials. Nevertheless, in Virtual Lab, many equipment models contain several materials to display various colors. The second method must be used to display an outlined effect on an object that contains multiple materials. Applying a customized Silhouette-Outlined shader can outline an object without changing all the color values of its materials. In addition, a variable timer in the script is required to create a glowing effect. The pseudocode of the algorithm is shown in Fig. 34.

```
function Glowing Effect (Time timer)
      timeInterval ← 1.0
      alpha ← 1.0
      timer ← Time.deltaTime
      if timer > timeInterval then
            timer ← timeInterval - timer
            alpha ← alpha - timer / timeInterval
      end if
      if alpha <= 0 then
            alpha ← timer / timeInterval
      else if alpha >= 1.0f then
            alpha ← timer / timeInterval - alpha
      end if
```

Fig. 34. Pseudocode for Glowing Effect.

When considering multi-platform game development, Unity is one of the best game engines for developers. Most of Unity's API and project structure are identical for all supported platforms and in some cases a project can simply be rebuilt to run on different devices. However, fundamental differences in the hardware and deployment methods mean that some parts of a project cannot be deployed between platforms directly. One example of different behavior between platforms is the input methods offered by the hardware. The *Input.GetAxis* function is frequently used on desktop platforms as a way of consolidating keyboard and joystick input. However, this function cannot be used for the mobile platforms directly because mobile platforms rely on touchscreen input. Therefore, it is necessary to design separate input functions for various platforms.

For example, a single touch in Android devices on the screen is treated as a left click and the *Input.mousePosition* property gives the position of the touch on if the finger is touching the screen. This means that games with simple mouse interaction can often work correctly between the desktop and mobile platforms. However, a desktop application can make use of more than one mouse button and a mobile application can detect multiple touches on the screen at a time. When using API, the problem can be managed partly by representing input with logical values that are then used by the rest of the application code. For example, a pinch gesture to zoom on a mobile device might be replaced by a plus/minus keystroke on the desktop; the input function could simply return a float value specifying the zoom factor. Moreover, it might be possible to use a two-finger tap on a mobile to replace a right button click on the desktop. Furthermore, if the properties of the input device are an integral part of the application, then it may not be possible to remodel them on a different platform. This may mean that application cannot be deployed at all or that the input and/or gameplay need to be modified extensively [36].

5.6 User Interface Design

Graphic system design is intensive and focuses on providing users with comprehensive view of the 3D virtual environment. The main objective of the graphic system design is to enable complete first-person viewing of the lab environment and process in the application, while simultaneously enhancing the user experience via visually appealing graphic design. This graphic system design involves lab environment design and UI menu design. To accurately mimic the settings in the real lab environment, the author created a sophisticated scene, composed of a 3D virtual lab and six 3D human models with animations. The illustration is shown in Fig. 35.

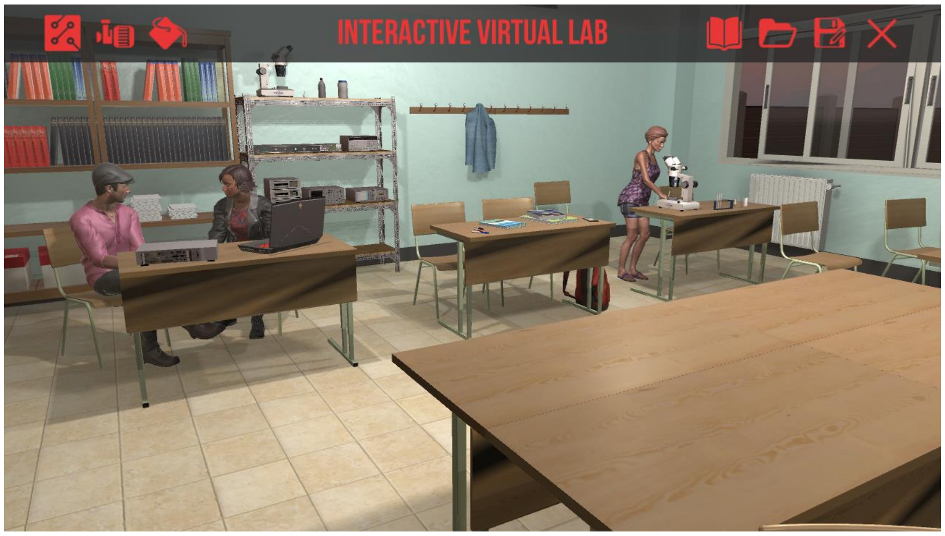


Fig. 35. Animated 3D Human Models in App Scene.

The UI menu design was based on five key principles [26]. The structure principle dictates that the UI is organized in a meaningful and useful way centered on clear, consistent models that are apparent and recognizable to users. A good UI design should separate dissimilar items and group similar items together. Next, the simplicity principle, says that UI design should provide users with simple, clear content and straightforward shortcuts that are meaningfully related to longer procedures. Thirdly, the visibility principle decrees that essential options and materials for a given task must be visible without distracting the user with extraneous or redundant information. Fourthly, the feedback principle states the design of UI should inform users with actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users. Lastly, the reuse principle maintains that ideally UI should be able to reuse internal and external components and actions, maintain consistency with purpose rather than arbitrary consistency, and therefore reduce the need for users to remember. Adhering to the five aforementioned principles, the finished UI in Virtual Lab is shown in Fig. 37 and Fig. 38.

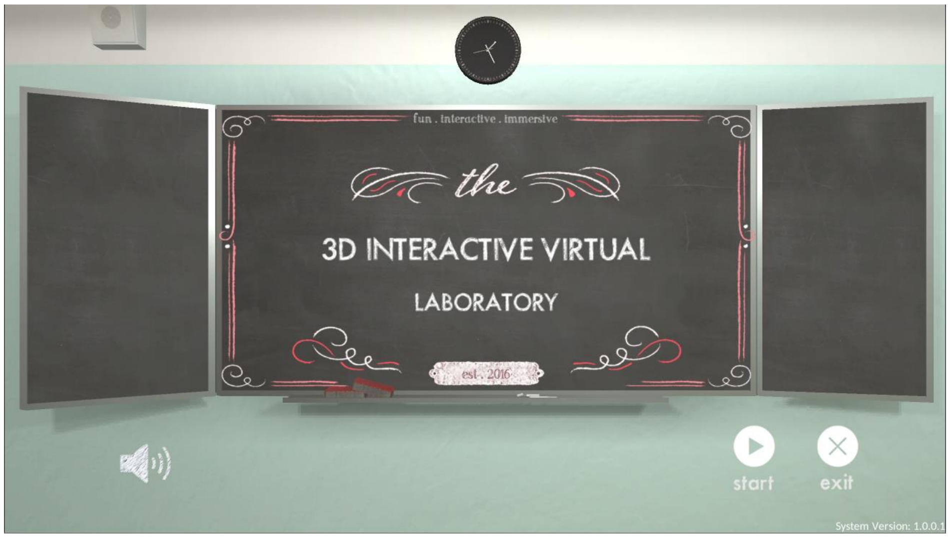


Fig. 36. UI Menu in Start Scene.

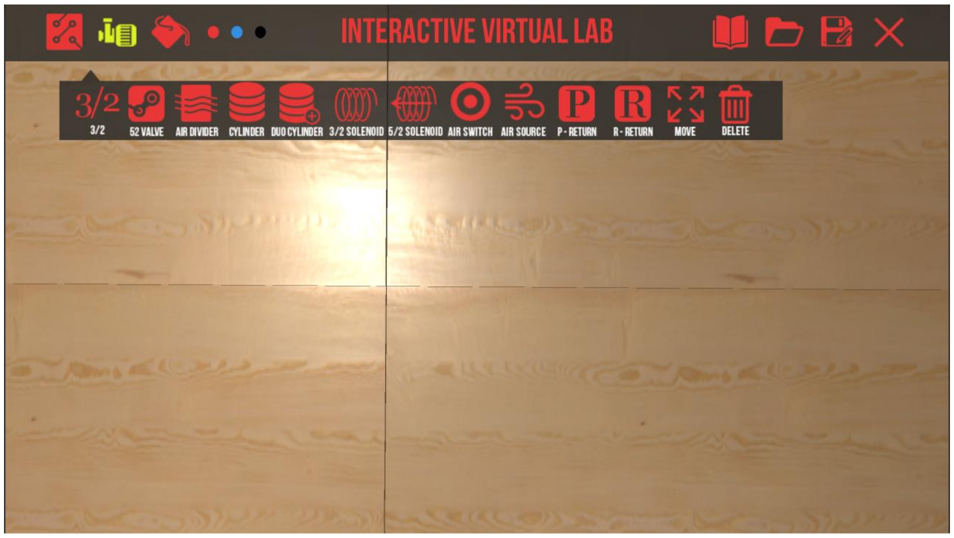


Fig. 37. UI Menu in App Scene.

Additionally, UI on different platforms should also be modified to improve user experience. In Virtual Lab, the author added four additional buttons for Android platform to provide users with the abilities of view pan, view rotate, operation confirm and operation cancel. Also, the graphics on Android devices are optimized to have the best performance, including compressing 3D model textures, removing light probes in the virtual environment, resizing the UIs for all the scenes, and so on. Fig. 38 shows screen capture of Virtual Lab running on an Android device.

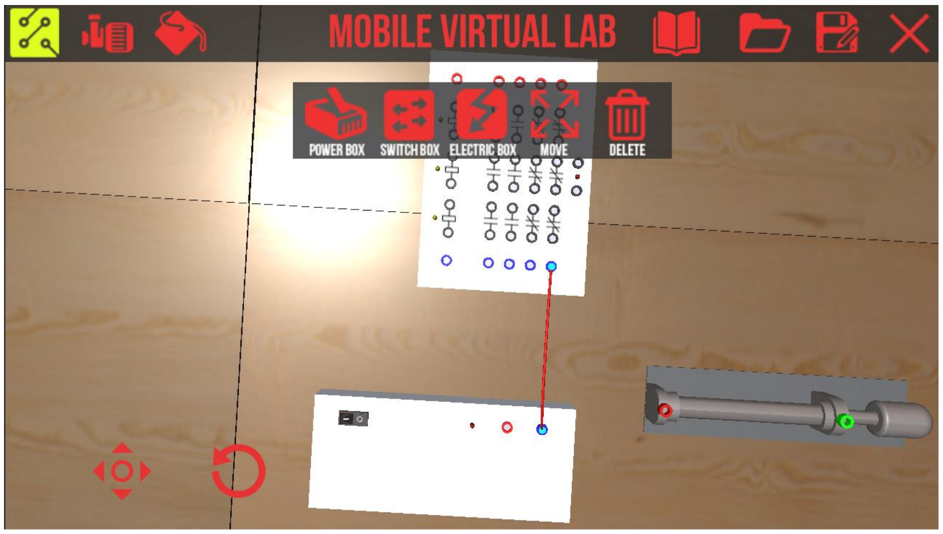


Fig. 38. Virtual Lab on Android Platform.

5.7 Serialization and Deserialization

Serialization is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database, or a file. The main purpose of using this technique is to save the state of an object and recreate it when needed. The reverse process is called deserialization [37]. There are two types of serialization: binary serialization and XML serialization. In binary serialization, all members in the serialized field can be serialized, and the overall serialization performance is enhanced. XML serialization provides more readable code as well as greater flexibility of object sharing and usage for interoperability purposes. After comparing two types of serialization methods, the author selected binary serialization for its faster and more efficient performance. When designing the serialization function in Virtual Lab, the states of both virtual equipment and virtual wires must be saved in order to recreate the same scenario if it is needed again in the future. However, due to the large number of data information contained in these items, serializing every piece of information into the database can be complex and time consuming. To tackle this problem, the author created a class called *SerializedItem*, which contains an array of partial information of equipment and wires to reduce the amount of serialization process. The partial information of equipment includes its three-dimensional world position and its current index number. The partial information of wires is comprised of the three Bezier curve positions, the current material color, and the neighbor nodes information. After users click on the save button, all the information above will be saved into the saved class, and the binary data will be written into a file. When de-serializing the saved class, the application will first extract the information of all the equipment and place them in the virtual environment, and then extract the information of all the wires and construct the relationship between the existing nodes. After finishing the deserialization process, the same virtual experiment is generated in the Virtual Lab.

5.8 Class Description

The detailed description of major classes in Virtual Lab is listed in TABLE 5.

TABLE 5

MAJOR CLASSES IN VIRTUAL LAB

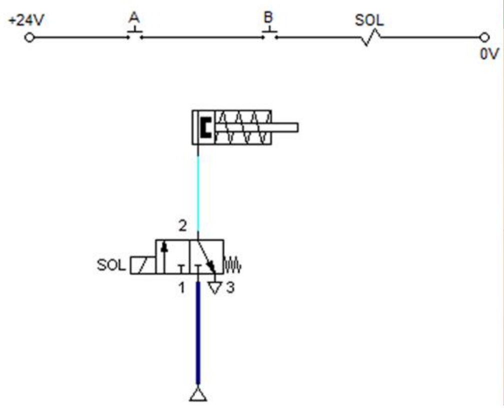
Class Name	Description
Singleton	It is used to restrict the instantiation of any derived class to one virtual object, such as GameController class and CameraController class
LabEquipment	It contains data and functions that can be inherited from another subclass. It is used to simplify and improve the quality of the code
CameraController	It is used to control the movement of the user view and it can also send execution command to the pointed virtual object
InputManager	It is used to provide users with the capability of interacting with the virtual objects
Node	It contains the voltage information of its own and its neighbor nodes. It is used to set the reference of a connector and it has a recursive algorithm to update the voltage information of its own and its neighbor nodes
LineController	It includes the voltage information of two connected nodes. It is also used to modify the material color of a wire
EquipmentController	It is used to track and modify the collision detection between different virtual equipment
GameController	It is used to control and manage various system events (UI system event and input system event) and it can also track and transmit data to other class
WireManager	It is used to generate virtual wires in the application
GameAudioController	It is used to turn on/off the audio source in the application
SerializedItem	It contains the data information of virtual equipment and virtual wires. It allows a user to save and load the progress of his/her virtual experiment

PowerBoxController, ElectricSwitchBoxController, ManualSwitchBoxController, AirSupplyController, AirSwitchController, FiveTwoSolenoidController, FiveTwoValveController, PneumaticReturnController, SpringReturnController, SingleCylinderController, DoubleCylinderController, ThreeTwoController, ThreeTwoSolenoidController	These class are derived class which inherit the data and functions of LabEquipment class. They are used to instantiate various virtual equipment and position them in the 3D world.

CHAPTER 6

RESULTS

This section shows the simulation results of Virtual Lab for typical programmable control logic encountered in introductory automation and control courses. It contains the ladder diagrams of six logic circuits and their corresponding wiring in Virtual Lab. Ladder diagrams are graphical diagrams based on the circuit diagrams of relay logic hardware and they are used for programmable logic controllers in industrial control applications. The correctness of PLC logic simulation in Virtual Lab has been validated.



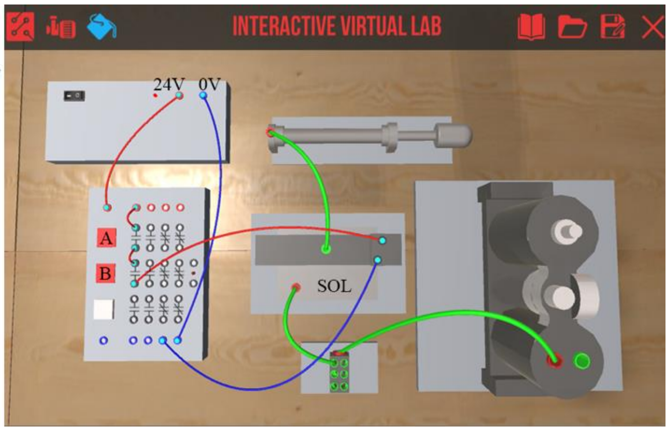


Fig. 39. AND Gate Ladder Diagram and Wiring Validation.

In Fig. 40, when button A and button B are being pressed, the 5/2 Solenoid can transmit the air to the Single Cylinder.

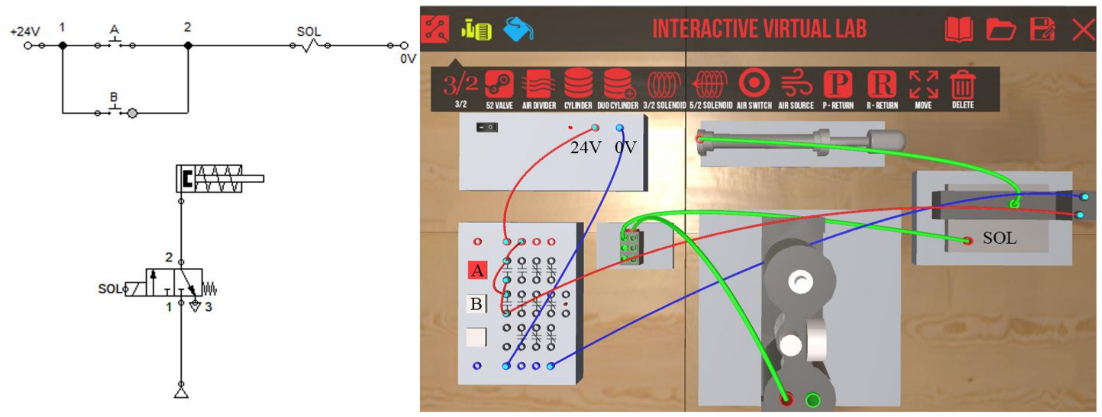


Fig. 40. OR Gate Ladder Diagram and Wiring Validation.

In Fig. 41, when button A or button B is being pressed, the 5/2 Solenoid can transmit the air to the Single Cylinder.

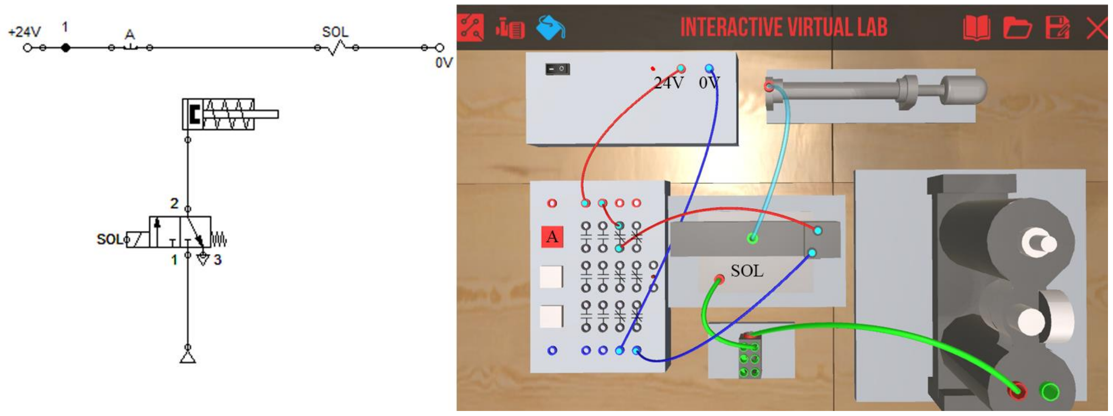


Fig. 41. NOT Gate Ladder Diagram and Wiring Validation.

In Fig. 42, when button A is being pressed, there is no air coming out from the 5/2 Solenoid, when button A is being pressed, the 5/2 Solenoid can transmit the air to the Single Cylinder.

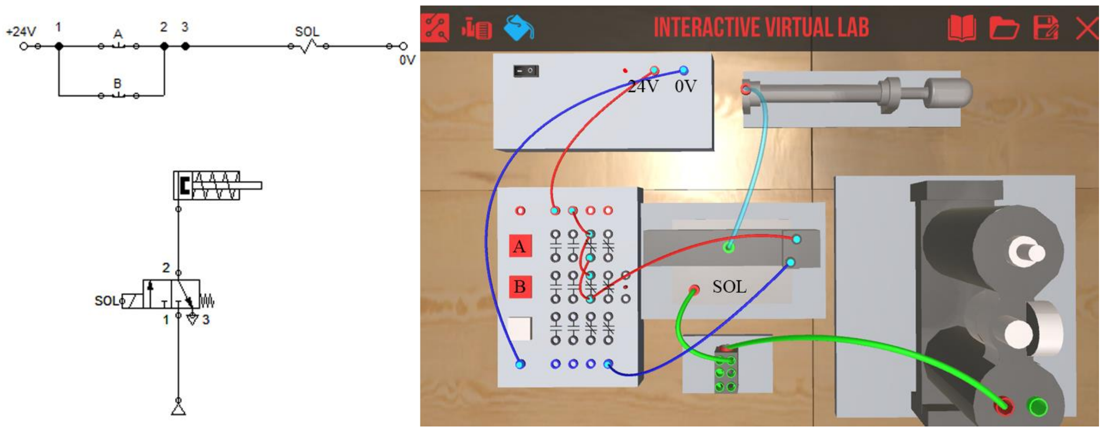


Fig. 42. NAND Gate Ladder Diagram and Wiring Validation.

In Fig. 43, when button A or button B is not being pressed, the 5/2 Solenoid can transmit the air to the Single Cylinder.

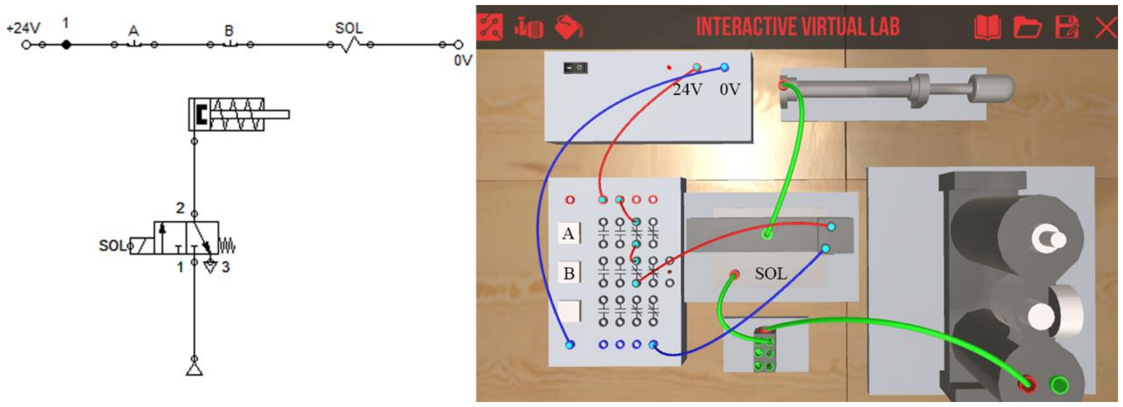


Fig. 43. NOR Gate Ladder Diagram and Wiring Validation.

In Fig. 44, when both button A and button B are not being pressed, the 5/2 Solenoid can transmit the air to the Single Cylinder.

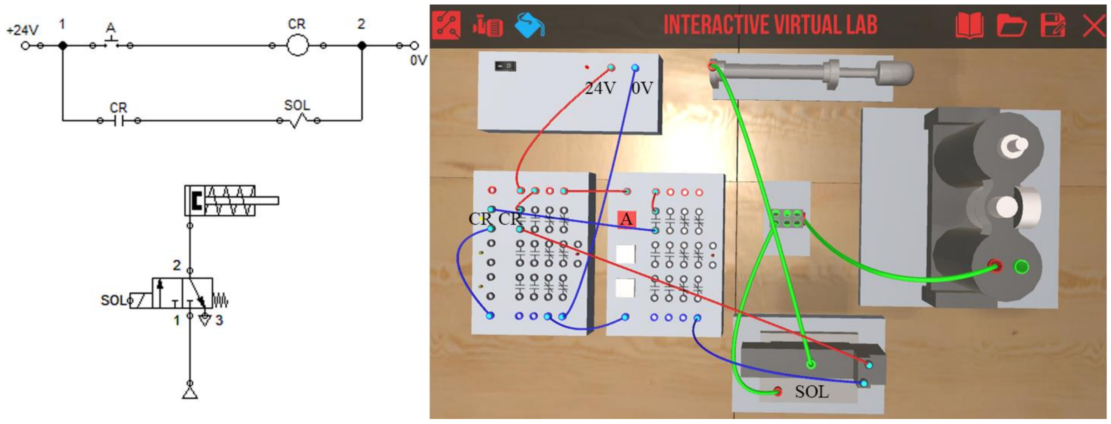


Fig. 44. Relay Control Ladder Diagram and Wiring Validation.

In Fig. 45, when button A is being pressed, the 5/2 Solenoid can transmit the air to the Single Cylinder.

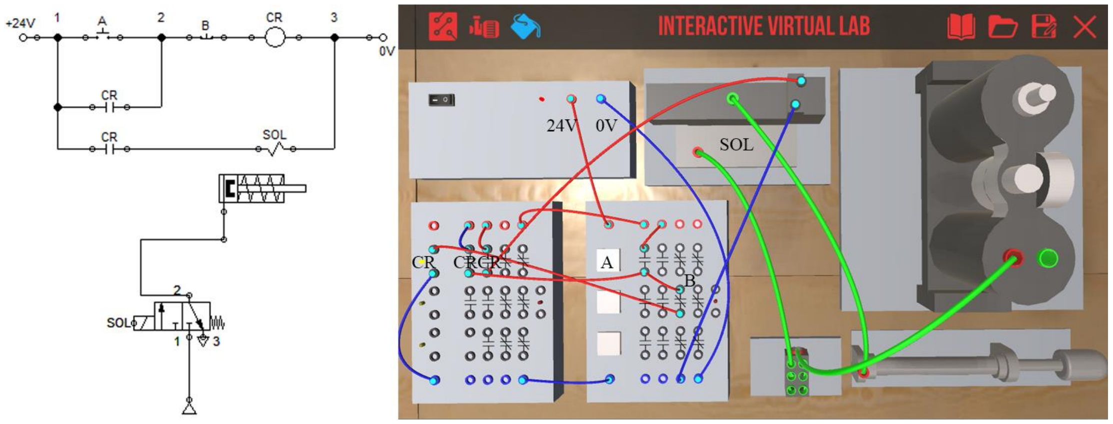


Fig. 45. Seal-in Circuit Control Ladder Diagram and Wiring Validation.

In Fig. 46, after the first-time button A is being pressed, even button A is held up, the 5/2 Solenoid can still transmit the air to the Single Cylinder.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This thesis presented the design, implementation, and results of Virtual Lab, a simulation application for complementing programmable control logic using electrical and pneumatic components. This application is an updated, advanced version of a previous implementation of Virtual Lab. The development of the original Virtual Lab was motivated by two elements. First, current online education in general cannot adequately provide students with needed practical, hands-on laboratory experience. Secondly, distance learning students who take MET 370/386 courses at Old Dominion University were disadvantaged by not having access to the lab facilities, compared with on-campus students.

After the first version of Virtual Lab was utilized in the course MET 370/386 Automation and Control offered by Old Dominion University in fall 2013, the overall feedback from students and the instructor was overwhelmingly positive. However, the Microsoft XNA Game Studio which was primarily used for developing Virtual Lab was no longer actively being updated. Unity is a leading game engine with a wide range of advanced capabilities and provides a natural choice for continuing development of Virtual Lab. This second version inherited all the functionalities from the previous one and has incorporated new features, such as pneumatic components, direct 3D drag and drop, and more flexible camera views, and a multi-system logic processor. Students can now use it to simulate programmable logic controller using combined electro-pneumatic experiments. Currently, Virtual Lab is still under active development, with plans to add more lab devices and equipment. Integration of the current project with other virtual reality devices such as Oculus Rift, HTC VIVE Samsung Gear VR, and Microsoft HoloLens is also being considered.

REFERENCES

[1] MIT, "MIT OpenCourseWare," ed: Massachusetts Institute of Technology, 2014.

[2] Harvard, "Harvard Open Courses: Open Learning Initiative," ed: Harvard University, 2014.

[3] edX, "edX Open Course," ed: edX, 2014.

[4] TEDTalks. *TED Talks*. Available: https://www.ted.com/talks

[5] DoD. *America's Army*. Available: http://www.americasarmy.com/

[6] B. I. Simulations. (2014, March 12). *Bohemia Interactive Simulations*. Available: http://armory.bisimulations.com/products/vbs2/overview

[7] DoD. (2014). *OneSAF*. Available: http://www.onesaf.net/community/index.php

[8] M. Liu, L. Belfore, Y. Shen, and M. Scerbo, "Uterine Contraction Modeling and Simulation," presented at the MODSIM World 2009 Conference and Expo, Virginia Beach, VA, 2010.

[9] Y. Shen and R. Pedada, "Automatic Generation of Texture Images for Wound Debridement Simulation," presented at the 2007 International Conference on Image Processing, San Antonio, Texas, 2007.

[10] Y. Shen, J. Seevinck, and E. Baydogan, "Realistic irrigation visualization in a surgical wound debridement simulator," (in eng), *Stud Health Technol Inform*, vol. 119, pp. 512-4, 2006.

[11] C. Warwick, "Everything you always wanted to know about SPICE* (*But were afraid to ask)," *EMC Journal*, no. 82, pp. 27–29, 2009.

[12] C. Y. Lin, Y. Shen, and M. Tomovic, "Hands-on Homework or Laboratory Development for Distance Learning Students in Programmable Logical Controller (PLC)," presented at the 121st ASEE Annual Conference Exposition, Indianpolis, IN, June 15-18, 2014.

[13] University of Illinois. *Buffon's Needle*. Available: https://mste.illinois.edu/activity/buffon/

[14] J. A. J. Andrew Sears, *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, 2nd ed. CRC Press, 2007.

[15] ComputerHope. *When was the first computer invented?* Available: http://www.computerhope.com/issues/ch000984.htm

[16] C. Chesher, "Colonizing Virtual Reality: Construction of the Discourse of Virtual Reality," *Cultronix,* vol. 1, no. 1, 1994.

[17] S. Choi, K. Jung, and S. D. Noh, "Virtual reality applications in manufacturing industries: Past research, present findings, and future directions," *Concurrent Engineering,* vol. 23, no. 1, pp. 40-63, 2015.

[18] G. Kaur, *VHDL: Basics to Programming*, 1st ed. Pearson, 2011.

[19] P. Randhawa and V. Shanthagiri, "Concept of Operations to System Design and Development-An Integrated System for Aircraft Mission Feasibility Analysis Using STK Engine, Matlab and Labview," *International Journal of Instrumentation and Control Systems (IJICS),* vol. 5, no. 4, 2015.

[20] Y. Altman. (2010). *Java-to-Matlab Interface*. Available: http://undocumentedmatlab.com/blog/jmi-java-to-matlab-interface

[21] ODU, "Mechanical Engineering Technology Online Course," ed: Old Dominion University.

[22] Y. Shen, C. Y. Lin, and M. Tomovic, "Software Design of a Virtual Lab Environment for Engineering Education."

[23] R. Lander. (2017). *Announcing the .NET Framework 4.7*. Available: https://blogs.msdn.microsoft.com/dotnet/2017/04/05/announcing-the-net-framework-4-7/

[24] M. J. Williams. (2012). *How to Learn XNA*. Available: https://gamedevelopment.tutsplus.com/articles/how-to-learn-xna--gamedev-150

[25] D. Takahashi. (2014). *John Riccitiello sets out to identify the engine of growth for Unity Technologies (interview)*. Available: https://venturebeat.com/2014/10/23/john-riccitiello-sets-out-to-identify-the-engine-of-growth-for-unity-technologies-interview/

[26] Unity. *Unity User Manual*. Available: https://docs.unity3d.com/Manual/index.html

[27] AutoDesk. *Polygon Modeling*. Available: https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-7941F97A-36E8-47FE-95D1-71412A3B3017-htm.html

[28] P. Schneider. (2014). *NURB Curves: A Guide for the Uninitiated*. Available: http://www.mactech.com/articles/develop/issue_25/schneider.html

[29] AutoDesk. *Convert NURBS surfaces to a polygon mesh*. Available: https://knowledge.autodesk.com/support/maya/learn-

explore/caas/CloudHelp/cloudhelp/2016/ENU/Maya/files/GUID-CE7DD4E2-0D14-4980-ADDA-2216735FF63B-htm.html

[30] C. Alexander and M. Sadiku. (2004). *Fundamentals of Electric Circuits*.

[31] D. J. Griffiths, *Introduction to electrodynamics*. Prentice Hall, 1999.

[32] R. Spence, *Introductory Circuits*. John Wiley and Sons, 2008, p. 99.

[33] C. Gonzalez. (2015). *What's the Difference Between Pneumatic, Hydraulic, and Electrical Actuators?* Available: http://www.machinedesign.com/linear-motion/what-s-difference-between-pneumatic-hydraulic-and-electrical-actuators

[34] TutorialsPoint. *Data Structure - Graph Data Structure*. Available: https://www.tutorialspoint.com/data_structures_algorithms/graph_data_structure.htm

[35] MicrosoftMSDN. *Microsoft Developer*. Available: https://msdn.microsoft.com/en-us/library/aa289148(v=vs.71).aspx

[36] UnifyCommunity. *Silhouette-Outlined Diffuse*. Available: http://wiki.unity3d.com/index.php?title=Silhouette-Outlined_Diffuse

[37] Ambysoft. *User Interface Design Tips, Techniques, and Principles*. Available: http://www.ambysoft.com/essays/userInterfaceDesign.html

VITA

Zelin Zhu

Education

M.S. Modeling and Simulation
2017, Old Dominion University, VA, USA

M.E. Electrical and Electronics Engineering
2014, Old Dominion University, VA, USA

B.S. Electrical and Electronics Engineering
2012, Nanjing University of Posts and Telecommunications

Professional Experience

JAN 2014 – DEC 2015	City of Norfolk, Norfolk, VA, USA Mobile Application Developer Norfolk Recreation, Parks and Open Space
AUG 2014 – DEC 2016	Old Dominion University, Norfolk, VA, USA Graduate Research Assistant Department of Modeling and Simulation

Research Projects

SEP 2016 – DEC 2016	Microsoft HoloLens Construction Simulation
JAN 2016 – DEC 2016	3D Virtual Interactive Lab
MAY 2016 – JULY 2016	Evacuation Simulation
MAR 2016 – MAY 2016	3D Interactive Mobile App for Jet-force Lab
FEB 2015 – NOV 2015	Cannonball Trail App

Publications

Z. Zhu, Y. Shen, K. Moberly, A. Santo, J. Owens, “Cannonball Trail: A Historical Tourism Mobile App”, *10th MSVE Student Capstone Conference*, pages 32-35, April 2016.

Z. Zhu, S. Ren, Y. Shen, “3D Interactive Massage App”, *Spring Simulation Multi-Conference*, April 2015.