Lymphatic Voyage:
Communicating 4D Immune Cell Dynamics
and Lymph Node Architecture
using WebGL-based Animation and Interactivity

by
Li Yao

A thesis submitted to Johns Hopkins University
in conformity with the requirements for
the degree of Master of Arts

Baltimore, Maryland
March, 2017

© 2017 Li Yao
All Rights Reserved
Abstract

The sophisticated functions of the immune system result from tightly orchestrated cell movements within organized microenvironments, such as the lymph nodes. Novel imaging techniques, including intra-vital microscopy and tissue clearing methods have advanced the understanding of the 3D architecture and 4D (3D + time) cellular dynamics of lymph nodes. These powerful tools allow researchers to explore volume renderings of temporal cellular microenvironments. Yet, there is a communication gap between research results and audiences, such as students and the scientific community, who do not have access to 4D viewing software and may have difficulty interpreting raw data. Teaching lymph node architecture and immune cell dynamics at the cellular level is challenging due to the lack of visual teaching tools.

The purpose of this project is to develop a WebGL (Web Graphics Library) - based web application that is widely accessible and based on novel dynamic immune cell data. 3D animations and 3D interactive models were created, and a web application was coded and deployed. This web application allows audiences to go on a “Lymphatic Voyage” on both desktop computers and mobile devices, during which they can explore lymph node architecture and study didactic information that explains the animated cellular drama.

This project provides a novel cross-platform educational resource for instructors and graduate students in the field of immunology to explore 4D lymph node architecture. It also contributes to the field of biomedical communication through the development of an innovative workflow utilizing WebGL to augment the learning experience.

Li Yao
Chairpersons of the Supervisory Committee

Mark J. Soloski, PhD, Preceptor
Professor of Medicine, Division of Rheumatology, Johns Hopkins University, School of Medicine

Lydia Gregg, MA, CMI, FAMI, Department Advisor
Assistant Professor, Division of Interventional Neuroradiology and Department of Art as Applied to Medicine, Johns Hopkins University, School of Medicine

Ronald N. Germain, MD, PhD, Content Advisor
NIH Distinguished Investigator, National Institute of Allergy and Infectious Diseases, National Institutes of Health
ACKNOWLEDGMENTS

I would like to thank my department advisor, Lydia Gregg, for the guidance, advice, patience and encouragement throughout the development of this thesis project. Her suggestions were always helpful in decision making and problem solving. I was very glad to work with such an energetic and intelligent advisor.

I would also like to thank Dr. Mark J. Soloski for proposing this exciting thesis topic, constantly providing resources and feedback on this project, inviting his graduate students to participate in reviewing content and contacting Dr. Ronald Germain for collaboration. This project wouldn’t be possible without his support.

My sincere thanks also go to Dr. Ronald Germain and his team Dr. Andrea Radtke and Dr. Weizhe Li. I appreciate their kindness in sharing their unpublished 3D datasets. A special thanks goes to Dr. Juraj Kabat, an imaging specialist at the National Institutes of Health (NIH), who taught me how to use the scientific imaging program, Imaris.

I am very grateful for the opportunity to study in the Department of Art as Applied to Medicine. I cannot express enough gratitude to the faculty members and staff in the department: Corinne Sandone, Gary P. Lees, Timothy H. Phelps, David A. Rini, Jennifer E. Fairman, Juan R. Garcia, Ian Suk, Norman Barker, Sandra Gabelli, Ann R. Altemus, Donald F. Bliss, Michael Linkinhoker, Dacia Balch, and Carol Pfeffer. Their efforts in both teaching and caring for students have made being in this program a wonderful experience.

Many thanks to my lovely classmates, Eleanor Bailey, Katie Bergdale, Sarah Chen, Daniel Hermansen, Julia Lerner, and Nicholas Reback, for making our class a warm family, for being the best listeners, for dragging me to the gym every day and for teaching me so many things in every aspect of life.

Finally, I would like to thank my parents, Renzhi Yao and Guolian Li. They have supported me on every decision I have made in my life.
TABLE OF CONTENTS

Abstract .................................................................................................................. ii

Acknowledgments .................................................................................................. iv

Table of Contents .................................................................................................... v

List of Tables .......................................................................................................... viii

List of Figures ......................................................................................................... ix

List of Abbreviations .............................................................................................. xi

1. Introduction

1.1 Architecture and cell dynamics of the lymph node ............................................. 1

1.2 Three-dimensional optical imaging of the lymph node ...................................... 2

1.3 Current teaching materials for lymph node structure ........................................... 4

1.4 HTML5 and WebGL-based real-time 3D visualization ......................................... 6

1.5 Objectives ........................................................................................................... 9

1.6 Audience .............................................................................................................. 9

2. Materials and Methods

2.1 Educational content goals and web application design ......................................... 10

2.2 Software overview ............................................................................................. 10

2.3 Creation of 3D models ......................................................................................... 13

2.3.1 Segmentation in Imaris .................................................................................. 13

2.3.2 Mesh optimization ......................................................................................... 15

2.3.3 de novo asset creation .................................................................................... 17

2.4 Pre-rendered animation ....................................................................................... 20

2.5 Interactive 3D models .......................................................................................... 20

2.5.1 Blender and Blend4Web .................................................................................. 20

2.5.2 Project initiation in Blend4Web ...................................................................... 20

2.5.3 Mesh optimization for web performance .......................................................... 21
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.4 Material creation in Node Editor</td>
<td>24</td>
</tr>
<tr>
<td>2.5.5 Lights and shadows in Blend4Web</td>
<td>24</td>
</tr>
<tr>
<td>2.5.6 Camera/viewport setup in Blend4Web</td>
<td>24</td>
</tr>
<tr>
<td>2.5.7 Annotation tracking in Blend4Web</td>
<td>27</td>
</tr>
<tr>
<td>2.5.8 Web application deployment and local testing</td>
<td>27</td>
</tr>
<tr>
<td>2.6 Web application user interface design</td>
<td>28</td>
</tr>
<tr>
<td>2.7 Web application development</td>
<td>28</td>
</tr>
<tr>
<td>2.7.1 Summary of web interactivity</td>
<td>28</td>
</tr>
<tr>
<td>2.7.2 Web architecture creation in HTML</td>
<td>29</td>
</tr>
<tr>
<td>2.7.3 Web style design in CSS</td>
<td>29</td>
</tr>
<tr>
<td>2.7.4 Web functionality and interactivity in JavaScript</td>
<td>29</td>
</tr>
<tr>
<td>2.7.5 Data storage in JSON</td>
<td>32</td>
</tr>
<tr>
<td>2.8 Version control and web application hosting</td>
<td>32</td>
</tr>
<tr>
<td><strong>3. Results</strong></td>
<td></td>
</tr>
<tr>
<td>3.1 Web application design</td>
<td>34</td>
</tr>
<tr>
<td>3.2 Three-dimensional models</td>
<td>43</td>
</tr>
<tr>
<td>3.3 Deployed web application</td>
<td>47</td>
</tr>
<tr>
<td>3.4 Access to assets</td>
<td>58</td>
</tr>
<tr>
<td><strong>4. Discussion</strong></td>
<td></td>
</tr>
<tr>
<td>4.1 Effectiveness of the interactive web application</td>
<td>59</td>
</tr>
<tr>
<td>4.2 Performance of the web application</td>
<td>59</td>
</tr>
<tr>
<td>4.3 3D model segmentation</td>
<td>60</td>
</tr>
<tr>
<td>4.4 Mesh optimization</td>
<td>60</td>
</tr>
<tr>
<td>4.5 WebGL frameworks</td>
<td>61</td>
</tr>
<tr>
<td>4.6 Responsive design</td>
<td>63</td>
</tr>
<tr>
<td>4.7 Implications for scientific and medical education</td>
<td>63</td>
</tr>
<tr>
<td><strong>5. Conclusion</strong></td>
<td>64</td>
</tr>
</tbody>
</table>
Appendix A: Informal Student Focus Group Questions .................................................... 65
Appendix B: Script for 3D Animation .................................................................................. 66
Appendix C: Annotations for 3D Exploration ..................................................................... 67
Cited References .................................................................................................................. 69
General References .............................................................................................................. 71
Vita ........................................................................................................................................ 73
LIST OF TABLES

Table 2.3.1 Immunostaining antibodies used on the mouse lymph node .................................................. 14
LIST OF FIGURES

Figure 1.1.1 Anatomy of the mouse lymph node ................................................................. 3

Figure 1.3.1 Immunostaining of lymph node cells using Ce3D ........................................ 6

Figure 2.1.1 Web application diagram .................................................................................. 11

Figure 2.1.2 Brief web application flowchart ...................................................................... 12

Figure 2.3.1 Immunostaining with different fluorescent-tagged cell subset-specific antibodies 14

Figure 2.3.2 Regions of interest manually drawn to segment the lymph node capsule .......... 14

Figure 2.3.3 Segmented 3D assets ......................................................................................... 15

Figure 2.3.4 Change in polygon structure for the lymph node capsule object .................... 16

Figure 2.3.5 Simplification: Clustering Decimation (Meshlab) .............................................. 17

Figure 2.3.6 Creation and optimization of blood vessels ...................................................... 18

Figure 2.3.7 Simplification: Quadric Edge Collapse Decimation (Meshlab) ....................... 18

Figure 2.3.8 Creation of reticular network ........................................................................... 19

Figure 2.5.1 Project Manager in Blend4web ........................................................................ 20

Figure 2.5.2 Settings used to create customizable web applications ................................... 22

Figure 2.5.3 Mesh and rendering comparison ...................................................................... 23

Figure 2.5.4 Settings for normal map baking in Blender .................................................... 23

Figure 2.5.5 Material setup in Node Editor (Blend4Web) .................................................... 25

Figure 2.5.6 Adding empty Plain Axes ............................................................................... 26

Figure 2.5.7 Camera settings in Blend4Web ....................................................................... 26

Figure 2.5.8 Annotation settings in Blend4Web ................................................................. 27

Figure 2.6.1 Exporting icons to SVG files from Adobe Illustrator for HTML5 usage .......... 28

Figure 2.7.1 The beginning section of the HTML file created for the web application ........ 30

Figure 2.7.2 An example of CSS code used to define visual properties of HTML elements ..... 30

Figure 2.7.3 The JavaScript function used to create events on the annotation tags .......... 31

Figure 2.7.4 A section of the JSON file that stored the data for the web application ............ 32

Figure 3.1.1 User interface design ...................................................................................... 34

Figure 3.1.2-3.1.9 Web application storyboard slides .......................................................... 35
Figure 3.1.10-3.1.13 Animation storyboard slides ................................................................. 39

Figure 3.2.1 Full mouse lymph node ......................................................................................... 44

Figure 3.2.2 Mouse lymph node with the capsule, subcapsular sinus and afferent lymphatic
vessel in cross section .............................................................................................................. 44

Figure 3.2.3 Lymph node vasculature ...................................................................................... 45

Figure 3.2.4 Lymph node medulla ......................................................................................... 45

Figure 3.2.5 B cell follicles ...................................................................................................... 46

Figure 3.2.6 T cell zones ........................................................................................................... 46

Figure 3.2.7 Red blood cells, B cell, T cell and dendritic cell .................................................. 46

Figure 3.2.8 Reticular network ............................................................................................... 47

Figure 3.3.1 Preloader screen .................................................................................................. 48

Figure 3.3.2 Introductory page with the interactive 3D model .................................................. 48

Figure 3.3.3-3.3.14 Animation scenes ..................................................................................... 49

Figure 3.3.15 Help page (desktop mode) ................................................................................ 55

Figure 3.3.16 Help page (mobile/touch mode) ........................................................................ 55

Figure 3.3.17 3D exploration page .......................................................................................... 56

Figure 3.3.18 3D objects (Capsule/Lymphatics) turned off ..................................................... 56

Figure 3.3.19 Close-up view of the model with displayed annotation description ..................... 57

Figure 3.3.20 Second canvas displayed .................................................................................... 57

Figure 3.3.21 Information panel ............................................................................................. 58
LIST OF ABBREVIATIONS

C4D: Cinema 4D
Ce3D: Clearing-enhanced 3D
CPU: Central Processing Unit
CSS: Cascading Style Sheets
DICOM: Digital Imaging and Communications in Medicine
DOM: Document Object Model
GPU: Graphics Processing Unit
HEV: High Endothelial Venules
HTML: HyperText Markup Language
JSON: JavaScript Object Notation
NIH: National Institutes of Health
NLA: Non-Linear Animation
SDK: Software Development Kit
SVG: Scalable Vector Graphics
URL: Uniform Resource Locator
VRML: Virtual Reality Modeling Language
WebGL: Web Graphics Library
XML: eXtensible Markup Language
1. INTRODUCTION

1.1 Architecture and cell dynamics of the lymph node

The immune system constantly monitors and protects the body from invading pathogens, internal toxins, and cancer cells. The circulatory lymphatic system is a vital part of the immune system and plays an important role in both innate and adaptive immune responses. Innate immune responses work early to control pathogens, recognize certain molecular patterns and induce non-specific defenses. Adaptive immune responses rely on lymphocytes, including T cells and B cells, to recognize and respond to a wide range of specific antigens from bacteria, viruses, and other disease-causing organisms (Murphy & Weaver, 2017a).

Lymphocytes encounter antigens in secondary lymphoid organs, such as lymph nodes, the spleen, and mucosal lymphoid tissues where they interact with antigen presenting cells and initiate adaptive immune responses (Murphy & Weaver, 2017b). Lymph nodes are located at intervals along the lymphatic vessels. Lymphatic vessels propel lymph fluid, absorbed from the interstitial fluid, to lymph nodes for antigen examination. Cellular components in the lymphatic system, such as lymphocytes and antigen presenting cells, are highly mobile, enabling them to collect cellular information throughout the entire body and quickly respond to unexpected pathogen invasion and tissue damage. During development, the vertebrate immune system generates up to $1 \times 10^9$ different clones (populations) of lymphocytes. Each clone has a single antigen receptor that confers unique antigen specificity. As a result, each antigen activates only a very small subset of the lymphocyte clones, among millions of different clones (Murphy & Weaver, 2017c).

If the immune system were to depend on random cellular movement to allow the interaction of antigens with rare antigen receptor bearing lymphocytes, the chances of generating a fast response to a specific antigen would be extremely low. Survival of an individual whose immune system can quickly respond to and successfully remove pathogens or cancer cells is much higher, and thus favored by evolution. Such selection has resulted in highly specialized tissues and systems that allow the controlled movement of immune cells, facilitating fast adaptive immune responses. Lymphocyte movements are strictly guided by molecular cues and organized
microenvironments. Specifically, in lymph nodes, anatomical features support efficient cell trafficking, cellular interactions for antigen recognition, and initiation of adaptive immune host defenses.

As shown in Figure 1.1.1, a lymph node has a fibrous capsule, beneath which is the subcapsular sinus. The two main compartments of a lymph node are the cortex and medulla. B cells are localized in B cell follicles, which make up the outer cortex, while T cells are distributed in the surrounding paracortex, also referred as the deep cortex or T cell zone (Qi et al., 2014). A network formed by stromal cells and collagen fibrils throughout the lymph node provides mechanical support and guides lymphocyte movements, facilitating efficient immune responses (Bajénoff et al., 2006). The lymph node has connections to both the cardiovascular system and the lymphatic system. Lymphocytes in the blood are attracted by chemokines to the lymph node. They squeeze through the walls of the specialized post-capillary blood vessels called high endothelial venules, and migrate on the reticular network to respective compartments in the lymph node (Bajénoff et al., 2006; Mionnet et al., 2011). Lymph, constantly drained from interstitial fluid, carries antigens in free form and in association with the antigen presenting dendritic cells and enters the lymph node through the afferent lymphatic vessels (Qi et al., 2014). Lymph flows from the subcapsular sinus, to the medullary sinuses and finally exits the lymph node via the efferent lymphatic vessel. A small amount of the lymph perfuses into the parenchyma and is absorbed by the blood vessels (Woodruff, Zawieja, Carroll, & Jr, 2015). Antigens are filtered and trapped in lymph nodes in this process. These and other intricate anatomical features of the lymph node create a highly organized environment that ensures each lymphocyte population is in close contact with the appropriate antigen presenting cells to initiate T cell and B cell activation.

1.2 Three-dimensional optical imaging of the lymph node

The complex tissue structure of the lymph node makes studying the spatiotemporal cellular events happening within it challenging. The development of novel imaging techniques has enabled scientists to obtain a better view of lymph node architecture and cellular dynamics.
**Figure 1.1.1 Anatomy of the mouse lymph node.** (A) The whole lymph node with its capsule intact. (B) Vasculature of the lymph node. (C) The capsule, subcapsular sinus and afferent lymphatic vessel are shown in cross section to expose internal structures of the lymph node.
Intra-vital microscopy is a technique that allows in vivo visualization of biological processes in live tissues or animals. This technique was used to document migration of lymphocytes from the blood vessels into the lymph node parenchyma, and cellular interactions on the reticular network (Bajénoff et al., 2006).

Tissue clearing methods, which make intact tissues or whole organs transparent by removing lipids, allow volume visualization of complex compartments through optical slicing instead of physical slicing (Chung & Deisseroth, 2013; Song et al., 2015). With the help of immunofluorescent staining that labels different cell populations using antibodies, spatial relationships of the cell populations in the tissue can be investigated without introducing distortion through slicing and reconstruction. Dr. Weizhe Li from Dr. Ronald Germain's lab at NIH has developed an advanced and optimized tissue clearing protocol, named Clearing-enhanced 3D (Ce3D). This has allowed visualization of the complex 3D structure of an entire mouse lymph node at a cellular level resolution for the first time (Unpublished).

1.3 Current teaching materials for lymph node structure

In the immunology program at Johns Hopkins University School of Medicine, an introductory course, Fundamentals of Immune Recognition and Response, is given to the first-year PhD and MD/PhD students by Dr. Mark Soloski. The course is intended to teach the students some basic topics relevant to the field of immunology, including lymph node architecture. Currently, information regarding this topic is presented in several different ways, including (i) text description, (ii) histological slides, (iii) intra-vital videos, (iv) volume-rendered 3D images/videos, (v) 2D illustrations, and (vi) 2D conceptual animations.

Feedback and suggestions regarding the teaching material specific to lymph node architecture was gathered from three PhD students who took the introductory course and two MD/PhD students who took a similar immunology course through an informal focus group held at the initial stage of this project (Appendix A). According to the students, the topic of lymph node architecture was one of the most complicated lectures in the introductory immunology course and it took them longer to study than other topics.
The graduate students noted numerous limitations to the current teaching materials. First, although histological slides show the real structure of lymph nodes under the microscope, each image is restricted to one cross-section. The intricate lymph node 3D structures, especially continuous compartments, such as blood vessels, cannot be captured in thesis slides. In addition, the graduate students indicated that reading histological slides was challenging because, having recently entered the field of immunology, they did not have much training doing this specialized task.

For both intra-vital videos and volume-rendered 3D structures, immunostaining that utilizes labeled antibodies to mark specific cell populations are performed to distinguish tissue compartments at the cellular level. Intra-vital videos demonstrate cell movements and interactions. According to the students, the videos were very helpful for them to appreciate the cellular events. However, since such experiments are performed on live animals, not all structures can be stained and shown using the antibodies.

Volume-rendered 3D structures presented in the lecture were pre-rendered, static (no cell movement) videos of small sections of tissue, which conveyed simple 3D structures. When it comes to complex structures or multiple overlapping structures, the volume rendering technique becomes limited at visualizing structures clearly. Cells shown in these volume renders are usually stained by antibodies, resulting in groups of cells visualized only by their surface, giving them a hollow appearance (Figure 1.3.1). When this group of complex hollow cells is presented as a volume rendering, the edges tends to be very cloudy and only simple structures can be easily interpreted. Therefore, even when scientists have 3D imaging datasets, they usually present them as 2D slides or pre-rendered 3D videos with only simple structures visualized. When the data is presented as pre-rendered 3D videos, the audience cannot change the view if they want to study different aspects of the model that are not included in the video. For scientists, if they want to present the 3D model from a different view, they have to render a new video. These were the issues that Dr. Ronald Germain and Dr. Weizhe Li encountered when presenting their Ce3D dataset. One solution would be to generate a surface representing each compartment of the lymph node by segmenting the 2D image stacks and allowing interactivity with the 3D model.
Lastly, the graduate students noted that the simplified 2D illustrations and 2D conceptual videos were very effective at conveying complex information. However, the 2D illustrations lack the capacity to depict 3D temporal and architectural information.

Understanding 3D structure of the lymph node is essential for understanding lymph node physiological functions and cell dynamics. Although many innovative methods, such as Ce3D, have been used to study spatiotemporal events in lymph nodes, there is no evidence in the literature that these datasets have been used in the creation of novel 3D educational material. There is a communication gap between research results and audiences, such as students and the general scientific community, who do not have access to 4D viewing software and may have difficulty interpreting raw data. Multiple audiences could benefit greatly from the development of web-based, interactive, 3D animations based on novel dynamic immune cell data.

1.4 HTML5 and WebGL-based real-time 3D visualization

Due to the advancement of computer graphics, many 3D visualization, reconstruction and manipulation tools have been developed for both the animation industry and the medical/scientific industry. Cinema 4D (C4D), Maya, Blender and ZBrush are popular 3D animation and modeling programs used in the animation industry. OsiriX and Horos have been used for
reconstructing and rendering radiological imaging datasets. Amira and Imaris enable researchers to visualize their 3D datasets generated from cell biology and neuroscience. All of these programs require local installation of the software and most of them are not free. In addition, these programs are not oriented to lay audiences because they are designed to customize 3D scenes and animate 3D models. For programs that are designed specifically for the audience to interact with 3D models, such as the Brain Explorer (“BrainExplorer”, 2017), a desktop application developed by the Allen Brain Institute, and Visible Body (“Visible Body”, 2017), a desktop/mobile application developed by Argosy Publishing, installation of the end user software on the local computer is required. To deploy the application on different operating systems, such as PC and iOS, and on mobile devices, such as iPads, android tablets and mobile phones, multiple versions of the software need to be developed separately using different programming languages. Because of the fast development of web browsers, which are widely used for daily tasks, most of these issues can be solved by switching from the development of stand-alone software and plugins to web-based platforms (Taivalsaari, Mikkonen, Anttonen, & Salminen, 2011).

Web browsers are accessible from almost all operating systems and devices, and have, therefore, become a new platform for rendering and presenting multimedia and 3D graphical content. The upgrade of HTML (HyperText Markup Language) to HTML5 and incorporation of WebGL (Web Graphics Library, a JavaScript application program interface) on all modern browsers allow users to view and interact with 3D models from any device, including desktop computers, tablets and mobile phones without using any plug-ins, such as Adobe Flash. HTML5 makes it possible for web browsers to natively include and handle multimedia and graphical content using the canvas tag. The WebGL release in 2012 by Khronos Group extended the capacity of the canvas feature to rendering 3D content. Unlike software-based rendering programs, WebGL uses the computer’s GPU (Graphics Processing Unit), thus preventing the overload of the CPU (Central Processing Unit), which is used by all other programs simultaneously, and boosts the rendering speed. Because all major browsers, including Google Chrome, Safari, Internet Explorer, Mozilla Firefox and Opera, currently support WebGL, and every modern computer and mobile device comes with a graphics card, WebGL-based 3D interactive applications can be built once and played everywhere. Because of these features and
many other performance advantages, HTML5/WebGL has been evolving quickly, resulting in higher quality graphical rendering and interactivity.

For educational purposes, HTML5/WebGL provides a powerful tool for sharing 3D interactive anatomical models and animations and making them widely accessible. Examples include: the molecule viewer window on the Protein Data Bank website (“Protein Data Bank”, 2017), pycortex WebGL MRI viewer (Gao, 2013), Brain Surface and Tractography Viewer (Ginsburg & Pienaar, 2011), bioWeb3D (Pettit & Marioni, 2013), Liver Anatomy Explorer (Pietzsch, Monch, Sommerfeld, Preim, & Preim, 2013), and iview: WebGL visualizer for protein-ligand complex (Li, Leung, Nakane, & Wong, 2014). Other scientific WebGL applications such as Experience Curiosity (Carnalla-Martinez & Ellison, 2017) also show the great potential of this new technique. Another advantage of HTML5/WebGL is the capability of embedding additional interactivity and functionality to the canvas element by introducing other HTML elements. For instance, HTML buttons can be added to manipulate 3D models, or trigger animations in the canvas. This allows audiences to control the pace of reading or watching the materials, and it aims to augment the learning experience.
1.5 Objectives

This project aimed to develop a widely accessible interactive 3D web application using HTML5/WebGL to allow audiences to go on a “Lymphatic Voyage” where they can explore and interact with the lymph node and study didactic information through the following objectives:

1. Communicate dynamic immune cell data to audiences through engaging, WebGL-based, 4D (time+3D) interactive educational media including an immersive animated microenvironment. Primary educational topics included (i) lymph node architecture, (ii) lymphocyte movement, (iii) lymph flow, (iv) B cell activation and (v) T cell activation.

2. Create didactic annotations explaining structural information and cellular events as the user explores the lymph node to provide an adaptive, self-paced learning experience.

1.6 Audience

Primary audience: Graduate and medical students who are taking the course, *Fundamentals of Immune Recognition and Response*, at Johns Hopkins University, School of Medicine.

Secondary audience: Other graduate and medical students interested in immunology, and scientists studying immunology who are seeking effective communicating tools.
2. Materials and Methods

2.1 Educational content goals and web application design

All materials related to lymph node architecture and dynamics that were taught in the graduate course Fundamentals of Immune Recognition and Response were studied in preparation for script creation and interactive design. Full versions of the animation script and annotation descriptions are available in Appendix B and Appendix C. Through in-depth discussions with content experts and graduate students, the following content creation objectives were established for this thesis project.

(i) Explain and teach the architecture of the lymph node and demonstrate how the structure facilitates immune cell movement, interaction and activation through guided pre-rendered 3D animation sequences in the Anatomical Tour mode of the web application.

(ii) Provide an interactive 3D Exploration mode in the web application that allows the audience to interact with the compartments of the lymph node and study the structures and functions of each compartment at their own pace.

A diagram of the web application (Figure 2.1.1) was designed as a guide for the web application development. The logic behind the web application used for coding was organized into a brief flowchart (Figure 2.1.2).

2.2 Software overview

Imaris® from Bitplane was used to analyze and segment the microscopic 3D dataset. Meshlab® was used to clean up the segmentation result and decimate the mesh. ZBrush® was used to fix holes in the mesh, optimize the segmented models, and create new assets for animation. Cinema 4D® (C4D) was used to further optimize the model geometry to create and render scenes for 3D animations. Adobe® Illustrator and Adobe® After Effects were used to compile and animate the 3D animations. Adobe Illustrator was used to create and export SVG (Scalable Vector Graphics) files for 2D assets in the web application. Blender® was used to create normal maps for the interactive 3D models.
Figure 2.1.1 Web application diagram.

Numbers indicate corresponding web app storyboard slides (Figure 3.1.2-3.1.9).
Figure 2.1.2 Brief web application flowchart.
Blend4Web Pro® was used to create materials, lighting, camera and annotation tracking positions to export to HTML5 canvas, and test WebGL contents. Sublime Text® was used to write codes for HTML5, CSS (Cascading Style Sheets), JavaScript and JSON (JavaScript Object Notation) files. BrowserSync® was used to locally test the web application. Git® and Github® were used for version control. Github Pages® was used to temporarily deploy the web application from a remote server.

2.3 Creation of 3D models

2.3.1 Segmentation in Imaris

A 3D imaging dataset of a wild type mouse lymph node provided by Dr. Weizhe Li contained a stack of 2D images generated using Ce3D method. It was converted from the Leica microscopic files (lif) to an Imaris project (ims). To reconstruct the 3D structure from the 2D image stacks, segmentation was performed in Imaris.

Imarix XT mode was selected in the Imaris software. Five antibodies were used to label different cell populations, shown in Table 2.3.1 and Figure 2.3.1. The cyan channel was used for B cell follicle segmentation. The green channel subtracted by the cyan channel (green - cyan) was used for T cell zone segmentation. The yellow channel was used for blood vessel segmentation. Merged blue and red channels (blue + red) were used for segmentation of the subcapsular sinus and medullary cords. Manipulation of channels, such as channel merging and subtraction were conducted using Plugin > Imaging Processing > Channel Arithmetics. Surface creation was initiated by selecting “adding new surface” for a selected channel(s). Segmentation was performed semi-automatically through interpolation between regions of interest manually drawn on different planes (Figure 2.3.2) (add new surface > skip automatic creation > add surface using magic wand > click mode), or performed automatically by the computer. The lymph node capsule (Figure 2.3.3 (A)) and B cell follicles were segmented semi-automatically, while the blood vessels (Figure 2.3.3 (B)), T cell zones and medulla were segmented automatically. After the surfaces were created, they were exported to VRML (Virtual Reality Modeling Language) format (Surpass > Export selected object to vrml) for future modification.
<table>
<thead>
<tr>
<th>Color code</th>
<th>Antibody</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Lyve-1-570</td>
<td>Lymphatic tissues</td>
</tr>
<tr>
<td>Green</td>
<td>CD8-488</td>
<td>T cells</td>
</tr>
<tr>
<td>Cyan</td>
<td>B220-510</td>
<td>B cells</td>
</tr>
<tr>
<td>Yellow</td>
<td>CD31-594</td>
<td>Blood vessels</td>
</tr>
<tr>
<td>Red</td>
<td>CD169</td>
<td>Macrophages</td>
</tr>
</tbody>
</table>

*Table 2.3.1 Immunostaining antibodies used to visualize the mouse lymph node.*

*Figure 2.3.1 Immunostaining with different fluorescent-tagged cell subset-specific antibodies. Text not intended to be read.*

*Figure 2.3.2 Regions of interest manually drawn to segment the lymph node capsule.*
2.3.2 Mesh optimization

The lymph node capsule object (Figure 2.3.4) demonstrates the change in polygon count during the mesh optimization process. The object exported from Imaris had 8,851,512 polygons (Figure 2.3.4 (A)). It was imported to Meshlab and Filters > Remeshing, Simplification and Reconstruction > Simplification: Clustering Decimation (Figure 2.3.5) was applied to simplify the geometry to 29,659 polygons (Figure 2.3.4(B)). After the decimation, the surface had some holes and was not smooth. So it was imported into Zbrush. Holes on the object were closed (Tool > Geometry > Modify Topology > Close holes), minor inflating (Brush > Inflate) was performed to fix artifactual flat planes and several applications of Dynamesh were performed to

Figure 2.3.3 Segmented 3D assets. (A) Capsule, (B) Blood vessels.

15
rewrap and optimize the mesh (Tool > Geometry > Dynamesh). The optimized object in Zbrush had 14,270 polygons (Figure 2.3.4(C)). The large scale geometry of the T cell zones, B cell follicles and medulla assets were generated through a similar workflow.

Figure 2.3.4 Change in polygon structure for the lymph node capsule object. (A) Object exported from Imaris (8,851,512 polygons). (B) Object after Simplification: Clustering decimation in Meshlab (29,659 polygons). (C) Object smoothed after Dynamesh in Zbrush (14,270 polygons).
Figure 2.3.5 Simplification: Clustering Decimation (Meshlab).

Optimization of the blood vessels, which had smaller and more complex geometry (Figure 2.3.6 (A)), was performed with a different workflow. The asset was divided into 11 parts because the geometry was too complex for Meshlab to be able to open the file (3.15GB). These 11 parts were merged in Zbrush into one object after optimization. Because the vessels in the capillary bed were very small and thin, a different simplification filter was used in Meshlab (Filters > Remeshing, Simplification and Reconstruction > Quadric Edge Collapse Decimation) (Settings shown in Figure 2.3.7). The structure of vessels reached cellular level, thus during the automatic segmentation, many capillaries were eliminated. Brush > CurveTube in Zbrush was used to reconnect the thinner vessels based on their directions (Figure 2.3.6 (B)).

2.3.3 de novo asset creation

The efferent and afferent lymphatic vessels and larger blood vessels were missing from the microscopic 3D dataset. Only small parts of the lymphatics that entered and exited the lymph node were visible. These assets were sculpted de novo in Zbrush and optimized using Dynamesh. The reticular network was dense and difficult to visualize without cross-staining other structures, therefore, it was not labeled in the dataset. However, some previous publications had images and short intra-vital videos of small areas of the reticular network (Bajénoff et al., 2006; Ma, Jablonska, Lindenmaier, & Dittmar, 2007). These publications were used as references for building the reticular network model.
Figure 2.3.6 Creation and optimization of blood vessels. (A) Object exported from Imaris that has holes and missing sections. (B) Object optimized after the described workflow.

Figure 2.3.7 Simplification: Quadric Edge Collapse Decimation (Meshlab).
To build the network, a Spline was created in C4D, deformed using two Displacers, and then cloned into a 4 by 4 array using a Cloner. A Random Effector was applied to the Cloner to randomize the position and rotation variables. The Cloner was duplicated twice and each duplicate was rotated slightly. All three Cloners were placed under a Metaball Object (Figure 2.3.8 (A)). Zbrush was used to sculpt and refine the preliminary object from C4D to generate a more accurate reticular network (Figure 2.3.8 B)).

*Figure 2.3.8 Creation of reticular network. (A) Preliminary object created in C4D. (B) Object refined in Zbrush. Text not intended to be read.*
2.4 Pre-rendered animation

All optimized meshes were imported into C4D for 3D animation. Materials, lights and camera were set up in C4D and animation scenes were rendered into PNG sequences. These PNG sequences were imported into Adobe After effects for editing, such as adding labels and changing transparency. The animation was exported and rendered using the Adobe After Effects Render Engine.

2.5 Interactive 3D models

2.5.1 Blender and Blend4Web

Blend4web, a WebGL framework and a 3D content creation suite based on the 3D engine Blender, was used in this project to embed interactive 3D models in the HTML canvas element. It is a Software Development Kit (SDK) tailored for 3D artists for creating web applications. The interactive 3D scenes were set up in Blend4Web.

2.5.2 Project initiation in Blend4Web

In order to produce a 3D web application that can be customized later (i.e. add additional HTML elements and JavaScript functions that manipulate the canvas element), a new project was created in the Blend4Web Project Manager. To access the Project Manager, Blender software was opened, Blend4Web mode was chosen, and a local host was opened through inputting “localhost: 6678” in the address bar of a web browser (Figure 2.5.1). The Project Manager was also used as a local server to test the in-progress web application, because the rendering results and interactivity can only be viewed when previewed on the web browser from this server before

![Blend4Web PRO 16.12.1 SDK](image)

*Figure 2.5.1 Project Manager in Blend4Web.*
it is deployed (Blend4web mode: 3D view window > Fast Preview). Settings shown in Figure 2.5.2 were chosen to create the customizable web application (Project Manager > Create a new project > Create project). A new folder with the blend/JSON/bin scene starter files and the customizable HTML file were created in the Blend4Web SDK/Projects folder. Additional CSS, Javascript and JSON files were created in the same folder. All other assets such as fonts and icons were saved in this folder as well.

2.5.3 Mesh optimization for web performance

In order to obtain better performance and make the web application accessible to more devices, the objects optimized from Zbrush were imported into Meshlab for additional decimation (Filters > Remeshing, Simplification and Reconstruction > Quadric Edge Collapse Decimation). Taking the T cell follicle asset as an example, the mesh polygon count decreased from 522,256 (Figure 2.5.3 (A)) to 32,640 (Figure 2.5.3 (C)).

Each polygon has a normal vector that tells the rendering engine how the light and material should be rendered. The rendering of high-polygon mesh is smoother than the low-poly because of its normal vectors (Figure 2.5.3 (B, D)). To simulate the normals of the high-polygon mesh on the low-polygon mesh, normals were edited in Blender. First, in the Tool panel of Blender, Smooth shading was selected. Then normal maps were baked from the high-polygon mesh and mapped to the low-polygon mesh. To bake a normal map, both the high-polygon and low-polygon meshes were imported into Blender. In the Blender Cycles Render mode, a Smart UV Map was created in the UV/Image Editor for the low-polygon mesh, and an Alpha Image was added. In the Material panel, a new Material was created and the Alpha Image was chosen as the Image Texture. The baking setting was in the Render tab of the Properties panel shown in Figure 2.5.4. To initiate baking, the high-polygon object was selected first and then the low-polygon object was selected. With both objects selected, the Bake function was initiated in the Baking panel. The baked normal map was then automatically saved to the Alpha Image that was created. This image needs to be saved before being assigned to the materials in the Blend4Web mode. Figure 2.5.3 (B) (D) and (F) show the rendering results of the high-polygon T cell zone, low-polygon T cell zone without the normal map and low-polygon T cell zone with the baked normal map.
Create a new project

<table>
<thead>
<tr>
<th>Setting</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Name (required)</td>
<td>lymphatic_voyage</td>
</tr>
<tr>
<td>Project Title (optional)</td>
<td>Lymphatic Voyage</td>
</tr>
<tr>
<td>Project Author / Company (optional)</td>
<td>Li Yao</td>
</tr>
<tr>
<td>Create Application Starter Files</td>
<td>Create</td>
</tr>
<tr>
<td>Create Scene Starter Files</td>
<td>Scene</td>
</tr>
<tr>
<td>Use Material Library</td>
<td>Use</td>
</tr>
<tr>
<td>Copy Project Management Script</td>
<td>Copy</td>
</tr>
<tr>
<td>Engine Binding Type</td>
<td>Web Player HTML</td>
</tr>
<tr>
<td>JavaScript Obfuscation Level</td>
<td>None</td>
</tr>
<tr>
<td>Web Player Params</td>
<td>Use</td>
</tr>
</tbody>
</table>

Figure 2.5.2 Settings used to create customizable web applications.
Figure 2.5.3 Mesh and rendering comparison. (A, B) High-polygon object (522,256 polygons). (C, D) Low-polygon object (32,640 polygons). (E, F) Low-polygon object with a normal map.

Figure 2.5.4 Settings for normal map baking in Blender.
2.5.4 Material creation in Blend4Web

In Blend4Web mode, materials were edited in the **Node Editor**. Each material started with a material core, which was modified and customized with additional variables. For instance, on the B cell follicle object (Figure 2.5.5 (A)), *glossy_plastic_material_core.001* that came with the Blend4Web material library was used as the material core. Three inputs were used to customize this material: **Color, Normal** and **Glossiness**. **Color** was defined by an **RGB** input node, the **Normal Map** was baked from the high-polygon object (mapped using the **Geometry** node) described in the previous section, and a **Value** node defined the **Glossiness**. Output variables from the material core were connected to the customized **Material** node and finally to the **Output** node. The **Normal Map** was connected to both the material core and the customized **Material** node because it was used in both nodes. A **UV map** of the low-polygon mesh was used to map the **Normal Map**. The components in *glossy_plastic_material_core.001* are shown in Figure 2.5.5 (B), which contains more material cores and nodes.

2.5.5 Lights and shadows in Blend4Web

To light the scene, one **Sun** light and five **Point** lights were placed in the scene. The **Sun** light propagates from an infinite plane in one direction without attenuation. It was used to light up the whole scene. The **Point** lights propagate from one point uniformly to all directions with gradual attenuation. They were used to create highlights on the model. For better performance, only the **Sun** light was used for shadow casting. To show the shadows on a certain object, **Properties > Object > Receive Shadows** was checked.

2.5.6 Camera/viewport setup in Blend4Web

The camera or the viewport, i.e., the canvas element, were set up in Blend4Web (**Properties > Camera**), but were also controlled by JavaScript. In Blend4Web, an empty **Plane Axes** (Figure 2.5.6) was setup at the center of the lymph node as the target for the camera. Another empty **Plane Axes** was setup as the start-up position for the camera. The camera limits for the start-up scene were set in the **Camera Panel** as well, shown in Figure 2.5.7. These variables restrain the camera movement style and limits in the **Viewport. Move Style** was set to **Target** to allow the camera to move around while looking at the target, in this case, the lymph node.
Figure 2.5.5 Material setup in Node Editor (Blend4Web). (A) Glossy material settings. (B) Components of the glossy material core. Text not intended to be read.
Figure 2.5.6 Adding empty Plain Axes.

Figure 2.5.7 Camera settings in Blend4Web.
**Distance Limits** were set to 50 to 250, to give the audience the opportunity to zoom in and out without losing control of the view through infinite zooming. **Rotation Limits** were not set in this application so that the audience can rotate the camera 360 degree. **Pivot Translation Limits** were not set so the audience can move the model in all directions with no limitations. **Use Panning Mode** was checked to let the audience pan the camera using the right click button. Auto-rotation of the camera in the web application was coded in JavaScript.

### 2.5.7 Annotation tracking in Blend4Web

Educational materials such as labels and annotations were included in the interactive 3D model. The annotation tags used in this web application are 2D HTML elements, which follow particular positions on the 3D model in the canvas element. To create a position in Blend4Web for the 2D annotation element to track, an empty **Plane Axes** was first created and moved to the designated position. In the **Object** panel (**Properties > Object**) of this **Plane Axes**, **Enable Anchor** was checked, **Custom Element** was chosen, and an **HTML Element ID** was assigned (Figure 2.5.8). The ID must be exactly the same as the 2D HTML annotation element for the system to correctly pair them. The element style was later customized in CSS.

**Figure 2.5.8 Annotation settings in Blend4Web.**

### 2.5.8 Web application deployment and local testing

When the web application was finished, the 3D model was exported to JSON and bin files from Blend4Web. The project was exported in the **Project Manager (Project Manager > deploy project)**. In this process, all the assets used in the web application were collected in the assets folder and all the SDK (Software Development Kit) JavaScript documents that were used in the application were combined into one concise JavaScript document.

The web application was tested on a local server using BrowserSync. To generate a local server and view the web application, the following script was used in the **Terminal** of iMac.

```sh
$ browser-sync start --server --directory --files "*"
```
2.6 Web application user interface design

Icons in the user interface were created in Adobe Illustrator and exported to XML (eXtensible Markup Language)-based vector image format SVG (Scalable Vector Graphics) for web usage. The export settings are shown in Figure 2.6.1. Mouse hovering, mouse in and mouse out styles of the icons and other user interface elements, such as text content, background color and buttons were coded in CSS.

![SVG Options](image)

*Figure 2.6.1 Exporting icons to SVG files from Adobe Illustrator for HTML5 usage.*

2.7 Web application development

2.7.1 Summary of web interactivity

The exported JSON and bin files from Blend4Web only contain the 3D model data. The web application interface and interactivity were coded in HTML, CSS, JavaScript and JSON. All codes were written using the code editor, Sublime Text. A brief flowchart that reveals the logic behind the JavaScript sources code can be found in Figure 2.1.2. Examples of code snippets are demonstrated below. To analyze the principles of functionality of this web application, see the source code and annotations at [https://github.com/LYAOI/Lymphatic_voyage_source_code](https://github.com/LYAOI/Lymphatic_voyage_source_code). The architecture of the elements in the web application was written in the HTML file, element styles were set in the CSS file, and functionality and interactivity were coded in JavaScript. JSON was used to store all the data that JavaScript needed to fill in the HTML elements.
2.7.2 Web architecture creation in HTML

Figure 2.7.1 shows the beginning part of the HTML file. The HTML file has two main elements, `<head>` and `<body>`. Title of the web application, character settings and viewport were set up in the `<head>` element. Fonts, CSS files and JavaScripts were linked in the `<head>` element as well. Other HTML elements that will be shown in the browser window, such as the canvas, annotation tags, and description text box were all placed in the division tags `<div>` within the `<body>` element. An ID name or class name was given to each element and used by CSS and JavaScript to add styles and functionality.

2.7.3 Web style design in CSS

A CSS file was used to define the style of each element pre-defined in the HTML or dynamically created later by JavaScript. For instance, in Figure 2.7.2, the division element with the ID “preloader_title” had a size of 790px by 80px. It was placed at the middle the window by aligning the element 50% from the left edge of the window with a -395px margin on the left (half size of the element), because the pivot point is at the element’s left top corner. The element’s opacity was set at 0 (“opacity: 0;”), so it will not be shown in the window when loaded and called out later through JavaScript. Text in this element was aligned horizontally in the center by setting “text-align: center;”.

2.7.4 Web functionality and interactivity in JavaScript

JavaScript shown in Figure 2.6.3 is the function used to create the annotation tags and bind the hovering and clicking events to these tags (`<div id=Annotation>` elements) in the HTML file. The empty Plain Axes placed in the Blend4web file that has the same anchor name as the HTML ID name were paired in the function. When the mouse is entering, leaving or clicking the annotation element, an event listener was bonded by using the “addEventListener” function. A description of the annotation was then created using “appendChild” and displayed on the screen using “show_interface_elem”. Most functions in this web application were appended or created dynamically and removed once the element was hidden from the window, so that the HTML DOM tree (Document Object Model) always has the minimum number of elements, which reduces the risk of overloading. Functions that define and call the second canvas were coded in JavaScript as well.
### Figure 2.7.1
The beginning section of the HTML file created for the web application.

```html
<!DOCTYPE html>
<html>
<head>
<title>Lymphatic Voyage: Virtual 3D tour</title>
<meta charset="utf-8">
<meta name="viewport" content="initial-scale=1.0, user-scalable=no, width=device-width">
<link href="https://fonts.googleapis.com/css?family=Raleway:200,300,400,500,600|Roboto:300" rel="stylesheet">
<link rel="stylesheet" href="lymphatic_voyage_dev.css" type="text/css">
<script src="lymphatic_voyage_dev.js"></script>
</head>
<body>
<!--canvas-->
<div id="main_canvas_container"></div>
<div id="second_canvas_container"></div>
<!--annotations-->
<div id="Annotation_1" class="room_anchor hidden"></div>
<div id="Annotation_2" class="room_anchor hidden"></div>
<div id="Annotation_3" class="room_anchor hidden"></div>
<div id="Annotation_4" class="room_anchor hidden"></div>
<div id="Annotation_5" class="room_anchor hidden"></div>
<div id="Annotation_6" class="room_anchor hidden"></div>
<div id="Annotation_7" class="room_anchor hidden"></div>
<div id="Annotation_8" class="room_anchor hidden"></div>
<div id="Annotation_9" class="room_anchor hidden"></div>
<div id="Annotation_10" class="room_anchor hidden"></div>
<div id="Annotation_11" class="room_anchor hidden"></div>
<div id="Annotation_12" class="room_anchor hidden"></div>
<div id="Annotation_13" class="room_anchor hidden"></div>
<div id="Annotation_14" class="room_anchor hidden"></div>
<div id="Annotation_15" class="room_anchor hidden"></div>
<div id="Annotation_2_1" class="inset_anchor hidden"></div>
<div class="dot"></div>
<div class="tag"></div>
</div>
<!--tour mode description-->
<div id="stage_description_cont"></div>
<!--explore mode description-->
<div id="explore_description_cont">
<div id="close_explore_descrip"></div>
</div>
<div id="close_inset_descrip"></div>
</div>
</body>
</html>
```

### Figure 2.7.2
An example of CSS code used to define visual properties of HTML elements.

```css
div#preloader_title{
    position: absolute;
    left: 50%;
    top: 50%;
    opacity: 0;
    width: 700px;
    height: 800px;
    margin-top: -168px;
    margin-left: -395px;
    text-align:center;
}
```
// create annotation anchors

function create_annotation_anchors() {
    var enter_action = "mouseenter";
    var leave_action = "mouseleave";
    var event = "click";

    if (is_touch()) {
        enter_action = "touchstart";
        leave_action = "touchend";
        event = "touchstart"
    }

    if (_cached_empty_objs)
        var empty_objs = _cached_empty_objs;
    else
        var empty_objs = _cached_empty_objs = m_scs.get_all_objects("EMPTY");

    for (var i = empty_objs.length; i--; ) {
        (function(){
            var empty_obj = empty_objs[i];

            if (m_anchors.is_anchor(empty_obj)) {
                var elem_id = m_anchors.get_element_id(empty_obj);
                var elem = document.getElementById(elem_id);

                if (_annotations[elem_id]) {
                    var elem_tag = elem.querySelector(".tag");
                    elem_tag.addEventListener(enter_action, add_hover_class);
                    elem_tag.addEventListener(leave_action, remove_hover_class);

                    elem_tag.addEventListener(event, function(){
                        while (_explore_description_cont.children[i])
                            _explore_description_cont.removeChild(_explore_description_cont.children[i]);
                    });

                    if (_stage_description_cont.style.display == "block"){
                        var explore_descip_text = document.createElement("div");
                        explore_descip_text.setAttribute("id", "explore_descip_text");
                        explore_descip_text.appendChild(generate_ex_descip_title(elem_id));
                        explore_descip_text.appendChild(generate_ex_descip_content(elem_id));
                        _explore_description_cont.appendChild(explore_descip_text);

                        show_interface_elem(_explore_description_cont);
                    }
                }
            }
        })();
    }
}

Figure 2.7.3 The JavaScript function used to create events on the annotation tags.
2.7.5 Data storage in JSON

To further optimize the performance and simplify the workflow, all data/text that were shown on the web application, were coded in JSON, a lightweight, platform-independent, data-interchange format (Figure 27.4).

![JSON code snippet](image)

*Figure 2.7.4 A section of the JSON file that stored the data for the web application.*

2.8 Version control and web application hosting

To keep track of the changes made in all the documents and back-up files for each step while working on this project, version control tool Git and Github.com were used. Folders were saved locally with Git monitor and remotely on Github.com as repositories. First, a repository was created on Github and a **remote repository URL** was generated. To initialize the local directory as a Git repository and push it to the remote repository at Github.com, the following steps were performed in the **Terminal** on iMac (codes are italicized):

```
$ git init
    #Initiate Git repository in current folder

$ git add .
    #Adds the files in the local repository and stages them for commit

$ git commit –m “First commit”
    #Commit the tracked changes
```
$ git remote add origin remote repository URL
   #Set up the new remote repository
$ git remote –v
   #Verify the new remote URL
$ git push –u origin master
   #Push the local repository to the remote repository

Every time new changes were made in the local repository, they were manually committed to Git and pushed remotely to Github using “git add”, “commit” and “push”. Github stores all previous versions while only leaving the most up-to-date version on the local repository. Repositories of all versions can be downloaded from multiple computers and changes in codes between different stages can be compared side by side on Github.com. In the remote repository, a Github Page was created to temporarily host the static HTML page.

In order to create a Github Page, the Master Branch was selected at the GitHub Pages section in the Settings page of the repository. The HTML page in the repository was renamed as index.html to be recognized and displayed on the Github Page.
3. Results

3.1 Web application design

The web application has 2 main modes: (i) Anatomical Tour and (ii) 3D Exploration. A series of icons was designed for this web application (Figure 3.1.1) and web app storyboards (Figure 3.1.2-3.1.9) were illustrated to plan the flow of the web application. For the Anatomical Tour mode (user interface shown in Figure 3.1.4), another set of storyboards was developed for the pre-rendered 3D animation sequences (Figure 3.1.10-3.1.13). Video and narration corresponding to each storyboard were listed on the left side of the animation storyboards.

![Image](image_url)

Figure 3.1.1 User interface design.
Figure 3.1.2 Web application storyboard slide 1.

Figure 3.1.3 Web application storyboard slide 2.
Figure 5.1.4 Web application storyboard slide 3. Pre-rendered animation storyboards are Figure 3.2.12-3.2.15.

Figure 5.1.5 Web application storyboard slide 4.
Figure 3.1.6 Web application storyboard slide 5.

Figure 3.1.7 Web application storyboard slide 6.
**Figure 3.1.8** Web application storyboard slide 7.

**Figure 3.1.9** Web application storyboard slide 8.
**Project:** Anatomical tour of lymph node

**Video:**
- rotation of transparent lymph node
- scale bar appears, cued to “10mm”

**Audio:**
Lymph nodes are secondary lymphatic organs with anatomic features that support initiation of adaptive immune host defenses. This model depicts the lymph node of a mouse, which is normally less than 10mm in length. Click the arrow to take a tour of the lymph node.

**Video:**
- T and B cell appear in insets
- highlight/show/hide different areas

**Audio:**
The lymph node is composed of the capsule, cortex and medulla. T cells are clustered in the T cell zones in the paracortex while B cells reside primarily in the follicles of the outer cortex. The medulla contains sinuses and medullary cords.

**Video:**
- arrows originate from arteriole and go to T cell zone and follicles
- zoomed in inset showing them exiting HEV

**Audio:**
T cells and B cells enter the lymph node through the blood supply and migrate to their respective zones.

*Figure 3.1.10 Animation storyboard slides 1-3.*
**Video:**
-show lymphocytes squeezing through HEV and reticular network

**Audio:**
The lymphocytes first squeeze through the wall of the specialized high endothelial venules (HEV), and exit from the junctions of the reticular network surrounding the HEV, and migrate on the reticular network.

**Video:**
-T and B cell appear in insets
-highlight/show/hide different areas

**Audio:**
The lymphocytes are guided by the reticular network and molecular cues. The reticular network is a supportive structure throughout the entire lymph node formed by stromal cells and collagen fibrils. It guides lymphocyte movements, and provides a crossroad for the interaction of rare clonal precursors of T and B cells with their cognate antigens.

**Video:**
-lymph flows from afferent vessel
-inset shows activated dendritic cells (veil cells)

**Audio:**
Lymph, which is drained from interstitial fluid, enters the subcapsular sinus through the afferent lymphatic vessels. It carries antigens in free form, and in the antigen presenting dendritic cells.

*Figure 3.1.11 Animation storyboard slides 4-6.*
Video:
- Lymph enters medullary sinuses from subcapsular sinus

Audio:
Most lymph drains into medullary sinuses and leaves the lymph node via the efferent lymphatic vessel. A small amount of lymph perfuses into the paracortex and is absorbed by blood vessels.

Video:
- Lymph exit through efferent vessel

Audio:

Video:
- Lymph diffuses into parenchyma and absorbed by blood vessels

Audio:

Figure 3.1.12 Animation storyboard slides 7-9.
Project: Anatomical tour of lymph node

**Video:**
- zoom out to show the whole transparent lymph node, with labels of different regions

**Audio:**
These intricate anatomical features of the lymph node create an organized environment that ensures each lymphocyte population is in close contact with the appropriate antigen presenting cells

**Video:**
- cell interactions on reticular network

**Audio:**

*Figure 3.1.13 Animation storyboard slides 10-11.*
3.2 Three-dimensional models

Several 3D models including the lymph node capsule, afferent vessels, B cell follicles, T cell zone, medulla, blood vessels, efferent vessel, T cells, B cells, dendritic cells and reticular network were created in Zbrush and C4D as described in Materials and Methods.

A series of 2D images were created to exhibit the anatomical details of the models (Figure 3.2.1-3.2.11). An additional still image was created to showcase the lymph node structures (Figure 1.1.1). The models for 3D animation were rendered in C4D, while the interactive 3D models were rendered in Google Chrome browser.
Figure 3.2.1 Full mouse lymph node.

Figure 3.2.2 Mouse lymph node with the capsule, subcapsular sinus and afferent lymphatic vessel in cross section.
Figure 3.2.3 Lymph node vasculature.

Figure 3.2.4 Lymph node medulla.
Figure 3.2.5 B cell follicles.

Figure 3.2.6 T cell zones.

Figure 3.2.7 Red Blood cells, B cell, T cell, and dendritic cell.
3.3 Deployed interactive web application

The web application “Lymphatic Voyage” was developed and deployed as an HTML, CSS and JavaScript package. The user interface, pre-rendered animations, interactive 3D models and functionality were all incorporated in this web application. Screenshots of the deployed interactive web application are shown in Figure 3.3.1-3.3.21.
Figure 3.3.1 Preloader screen. Corresponds to Figure 3.1.2.

Figure 3.3.2 Introductory page with the interactive 3D model. Corresponds to Figure 3.1.3.
Figure 3.3.3 Animation scene 1. Corresponds to Figure 3.1.10 storyboard slide 1.

Figure 3.3.4 Animation scene 2. Corresponds to Figure 3.1.10 storyboard slide 2.
3/6 T cells and B cells enter the lymph node through the blood supply and migrate to their respective compartments. The lymphocytes first squeeze through the walls of the specialized blood vessels called high endothelial venules, and migrate on the reticular network.

Figure 3.3.5 Animation scene 3, part 1. Corresponds to Figure 3.1.10 storyboard slide 3.

3/6 T cells and B cells enter the lymph node through the blood supply and migrate to their respective compartments. The lymphocytes first squeeze through the walls of the specialized blood vessels called high endothelial venules, and migrate on the reticular network.

Figure 3.3.6 Animation scene 3, part 2. Corresponds to Figure 3.1.10 storyboard slide 3.
Figure 3.3.7 Animation scene 3, part 3. Corresponds to Figure 3.1.11 storyboard slide 4.

Figure 3.3.8 Animation scene 4. Corresponds to Figure 3.1.11 storyboard slide 5.
Figure 3.3.9 Animation scene 5, part 1. Corresponds to Figure 3.1.11 storyboard slide 6.

Figure 3.3.10 Animation scene 5, part 2. Corresponds to Figure 3.1.12 storyboard slide 7.
Figure 3.3.11 Animation scene 5, part 3. Corresponds to Figure 3.1.12 storyboard slide 8.

Figure 3.3.12 Animation scene 5, part 4. Corresponds to Figure 3.1.12 storyboard slide 9.
These intricate anatomical features of the lymph node create an organized environment that ensures each lymphocyte population is in close contact with the appropriate antigen-presenting cells.

Figure 3.3.13 Animation scene 6, part 1. Corresponds to Figure 3.1.13 storyboard slide 10.

Figure 3.3.14 Animation scene 6, part 2. Corresponds to Figure 3.1.13 storyboard slide 11.
Figure 3.3.15 Help page (desktop mode). Corresponds to Figure 3.1.5.

Figure 3.3.16 Help page (mobile/touch mode). Corresponds to Figure 3.1.6.
Figure 3.3.17 3D exploration page. Corresponds to Figure 3.1.7.

Figure 3.3.18 3D objects (Capsule/Lymphatics) turned off.
**Figure 3.3.19** Close-up view of the model with displayed annotation description.

**Figure 3.3.20** Second canvas displayed. Corresponds to Figure 3.1.8.
Figure 3.3.21 Information panel. Corresponds to Figure 3.1.9.

3.4 Access to assets

Samples of the thesis can be accessed at liyaovisuals.com. The web application created from this thesis will be hosted online after further optimization and testing. Please contact liyaovisuals@gmail.com for more information. Part of the source code of the web application can be viewed at https://github.com/LYAO1/Lymphatic_voyage_source_code.

The author can also be contacted through the Department of Art as Applied to Medicine at Johns Hopkins University School of Medicine, http://medicalart.johnshopkins.edu/.
4. DISCUSSION

4.1 Effectiveness of the interactive web application

The primary goal of this thesis project was to create an effective teaching tool to help students learn complex lymph node 3D architecture and cell dynamics. An informal focus group was held during the preparation stage of this project to collect opinions about this topic from graduate students who have taken the course, *Fundamentals of Immune Recognition and Response*, taught at Johns Hopkins University School of Medicine. Survey questions can be found in Appendix A.

According to three PhD students who have taken this course, and two MD/PhD students who have taken a similar immunology course, understanding this part of the lecture was more difficult compared to other lectures due to the complexity of the 3D structure. This preliminary feedback is detailed in section 1.3.

In summary, they wished they had more illustrations and 3D models to clarify lymph node architecture. Their feedback indicates that a steep learning curve is associated with this subject. Suggestions from the graduate students were incorporated into the design of the web application and future meetings will be arranged to obtain feedback on the final product from the same group of students to address the effectiveness.

4.2 Performance of the web application

Since “Lymphatic Voyage” was built as a static HTML application, the website does not dynamically request data from the server side. Once the page is fully loaded and cached, the performance is solely dependent on the local GPU and CPU.

Several aspects should be considered when building 3D scenes to optimize the performance for WebGL. In Blend4Web, the number of light sources, number and quality of shadows, polygon count of the mesh, and ambient occlusion can all affect the performance. In this web application, minimum light sources were used, only one shadow was allowed and ambient occlusion was turned off. The mesh was also optimized as described in Materials and Methods. For materials
that do not require interactivity and real-time rendering, like the Anatomical Tour mode, pre-rendered animations have been created. In addition, HTML, CSS and JavaScript were minified for better performance utilizing several open-source online tools. (“Minify Resources (HTML, CSS, and JavaScript),” 2016). This optimization process significantly increased the performance. During testing, the remotely hosted web application ran smoothly on a 2015 iMac (Processor: 3.3 GHz Intel Core i5; Graphics: AMD Radeon R9 M395X 4096 MB) and a 2016 iPad Pro (A9X chip with 64-bit architecture with Embedded M9 coprocessor).

4.3 3D model segmentation

In medical education and communication, many 3D surfaces are generated through segmenting radiological DICOM (Digital Imaging and Communications in Medicine) datasets using OsiriX or Horos. The 3D cellular imaging datasets employed in this project were composed of a stack of 2D images, similar to DICOM files. These images were saved as an Imaris project. Imaris is a scientific image visualization program designed for managing cellular data. Due to the limited accessibility to Imaris, at the initial stage of the project, attempts were made to manage the cellular dataset in Horos. In Imaris, images of different channels were first exported separately to a single 2D image series. In Horos, this image series was converted to a DICOM dataset using a plug-in (Plug-in > Database > JPEG to DICOM). Semi-automatic segmentation can be done in Horos by manually drawing regions of interest. However, manipulation of different channels and automatic segmentation for an individual channel can only be performed in Imaris, because Imaris is specifically tailored for manipulating microscopic datasets. Because of the limitation of OsiriX/Horos, Imaris was ultimately used for segmentation for this thesis project.

4.4 Mesh optimization

The surfaces generated in Imaris resulted in assets with a high polygon count. The complexity of the mesh significantly affects the rendering time for both pre-rendered animation and the real-time rendered animation, therefore, optimization of the mesh was necessary.
To decrease the polygon count and smooth the assets for downstream workflow, several 3D programs were employed to optimize the assets individually to achieve better real-time rendering performance.

Meshlab is a free light-weighted software. It has multiple customizable decimation options. The Clustering Decimation function (Figure 2.3.5) decimates the mesh at a high strength and is suitable for meshes that do not need much surface detail preserved. The Quadric Edge Collapse Decimation can be customized to preserve the boundary of the mesh and thus keeps more detail.

Zbrush also has a Decimation Master plug-in, the percentage of decimation can be defined but it has fewer customizable variables than Meshlab. However, the Dynamesh function in Zbrush is a unique tool that optimizes meshes during sculpting. In addition, the resolution of Dynamesh can be defined to decrease or increase the polygon count.

Both C4D and Blender have Decimation function as a mesh Modifier. It is named Polygon Reduction in C4D and Decimate in Blender. Because they are modifiers, they do not change the original mesh until it is committed, thus the mesh can still be modified while decimation is applied. In C4D, mesh reduction strength, boundary curve preservation, polygon quality and mesh quality can be customized. In Blender, 3 modes (Collapse, Un-subdivide and Planar) can be chosen to perform the decimation.

In this thesis project, decimation was primarily performed in Meshlab because there were two decimation options each suitable for different assets.

4.5 WebGL frameworks

WebGL version 1.0 was released in March 2011 and version 2.0 was finalized in January 2017. Ever since it was released, people have been developing WebGL frameworks and libraries to facilitate the development of 3D interactive web applications (“List of WebGL frameworks”, 2016). Among all the WebGL frameworks, three.js is the most popular open-source framework. However, three.js is mostly dependent on scripting. The three.js editor (https://threejs.org/editor) is the user interface for creating 3D scenes and manipulating the camera, and it only has very basic functions. Three.js is more adapted for the computer science industry, in which scripts are
used to directly customize 3D models, materials, and lights. 3D game engines such as Unity and Blender, used by 3D artists for 3D modeling and animation, have also started to incorporate plugins to export to WebGL-based applications. These 3D game engine-based WebGL frameworks are friendly to artists because they have a well-developed user interface. Although Unity is a very popular 3D game engine, its WebGL exporter is not well-developed (Bezuhoff, 2015).

In Blender, a new mode called Blend4Web was developed to help artists build 3D scenes and export them directly to WebGL. Many interactive functions can be embedded on objects in 3D scenes without coding. Although the 2D elements in web applications and associated interactivity still need to be coded using JavaScript, it is much easier to learn. In addition, Blend4Web has a straightforward annotation function. Annotations that are important for educational web applications can be easily created. In this function, the 3D position is constantly translated to the 2D position in the browser window and sent to the corresponding annotation tag. The advantage of these 2D annotation tags over the annotation tags created in the 3D documents (3D objects) are: (i) they are vector and text based so they are crisp all the time, (ii) they are always facing the camera, (iii) they can be customized easily with CSS, JavaScript and JSON, (iv) they reduce the number of 3D models in a scene, making it faster. Apart from the artist-friendly user interface and customizable functions, Blend4Web also creates web applications with better performance compared to Unity in many different aspects (Prakhov, 2016). Blend4Web is a suitable and powerful program for 3D artists who wish to create 3D interactive web applications.

For this project, Blend4web was used to embed interactive 3D models in the HTML canvas element. It shares the same software user interface that Blender uses, but the functionalities are quite different between Blender and Blend4web, because Blend4web is completely based on JavaScript for browser-based real-time rendering. Blender has two default rendering engines: **Blender Render** and **Cycles Render**. Blend4web uses its own rendering engine. In addition, the materials for Blend4web can only be created using the **Node Editor**. Because of all these differences, many settings, especially for materials, from **Blender Render**, **Cycles Render** and other 3D software like C4D cannot be used in Blend4web, therefore the interactive 3D scene for this project had to be set up directly in Blend4Web.
4.6 Responsive design

Considering that students and instructors may need to access this web application from various devices, responsive design was included. Because of the complexity of the web application, the smallest size limit used for responsive design was the Ipad Pro 12.9 inch tablet. When a mobile device is detected, the web app starts to detect touching gestures and responds to them based on the JavaScript settings. To further optimize the application for smaller screen size, the layout of the HTML elements will be re-designed, and most of the elements will be expandable so that they do not block the canvas on a small screen.

4.7 Implications for scientific and medical education

The project created in this thesis has successfully incorporated both didactic strategies and novel 3D interactive techniques into teaching material for a complex scientific subject. Limitations still exist for such web applications. For example, the performance is dependent on the end user's device, and a web connection is required for accessing the resource. However, the wide accessibility (cross-platform) and interactivity provided by the web application cannot be replaced by any stand-alone software package. For scientific and medical education, which employs many forms of 3D data, interactivity of 3D models is in high demand. Such interactivity allows students to watch and read the educational material at their own pace, thus leading to a better learning experience.

For artists who wish to create 3D teaching material, real-time rendering simplifies the workflow by skipping the pre-rendering process. In other words, the 3D assets and scenes can be exported and viewed immediately. This workflow saves time if modifications must be made to the assets and scenes. Another advantage of web-based applications is that they can be set to automatically respond in real-time to dynamic datasets. This property can be utilized to effectively present newly published or updated scientific datasets, with minimum human input.
5. CONCLUSION

Teaching lymph node 3D architecture and cell dynamics is challenging because of the complexity of the subject matter. By combining cutting-edge 3D imaging datasets of the lymph node, didactic designs, and the newest WebGL-based real-time rendering techniques, a 3D interactive application “Lymphatic Voyage” was developed. Future work can be done to further enrich the educational materials and optimize the online performance of the web application.

Incorporation of innovative cross-disciplinary technologies into educational projects like “Lymphatic Voyage” opens the door to new possibilities in the field of biomedical communication. This novel web application provides a powerful new tool for science education and communication. In addition, the workflow described here provides a reference for the development of other 3D interactive web applications.
APPENDIX A: INFORMAL STUDENT FOCUS GROUP QUESTIONS

1. Without any additional work, how much (percentage) did you understand the lecture about the lymph node structure and function?

2. Which part(s) was hard to understand/follow? Why?

3. What material in the lecture slides helped you understand the subject the most?

4. How did you study for this subject after the lecture? What was the most helpful material? How did it vary from the way the material was presented in the lecture?

5. What do you wish you could have had to help you better understating this subject?

6. We are thinking about developing a visual tool to help students learn this subject. Rank the following by importance: animation with narration, 3D models and ability to rotate, ability to manipulate the structures (click, hide and show), ability to zoom in and out.

7. When you navigate through the conceptual animation scenes, would you like to have the ability to access the original lab data (literature)?

8. Is there anything we haven’t mentioned that you want to add?
APPENDIX B: SCRIPT FOR 3D ANIMATION

Lymph nodes are secondary lymphoid organs with anatomic features that support initiation of adaptive immune host defenses. This model depicts the lymph node of a mouse, which is normally less than 10mm in length. Click the arrow to take a tour of the lymph node.

A lymph node is composed of the capsule, cortex and medulla. T cells are clustered in the T cell zones in the paracortex while B cells reside primarily in the follicles of the outer cortex. The medulla contains sinuses and medullary cords.

T cells and B cells enter the lymph node through the blood supply and migrate to their respective compartments. The lymphocytes first squeeze through the walls of the specialized blood vessels called high endothelial venules, and migrate on the reticular network.

The movement of lymphocytes is guided by the reticular network and molecular cues. This reticular network is a supportive structure throughout the entire lymph node formed by stromal cells and collagen fibrils. It provides a crossroad for the interaction of rare clonal precursors of T and B cells with their cognate antigens.

Lymph, which is drained from interstitial fluid, enters the subcapsular sinus through the afferent lymphatic vessels. It carries antigens in free form, and in the antigen presenting dendritic cells. Most lymph drains into medullary sinuses and leaves the lymph node via the efferent lymphatic vessel. A small amount of lymph perfuses into the paracortex and is absorbed by blood vessels.

These intricate anatomical features of the lymph node create an organized environment that ensures each lymphocyte population is in close contact with the appropriate antigen presenting cells.
APPENDIX C: ANNOTATIONS FOR 3D EXPLORATION

1. Afferent lymphatic vessel

Lymph is supplied to lymph nodes through multiple afferent lymphatic vessels. Lymph circulation is controlled by (1) contraction of the lymphatic vessel wall and (2) compression of the lymphatic vessels by external forces (e.g. the contractions of skeletal muscles).

2. Capsule

The lymph node is surrounded by a fibrous capsule, beneath which is a sinus system lined by reticular cells, cross-bridged by fibrils of collagen and other extracellular matrix proteins. The sinus is filled with lymph, macrophages, dendritic cells and other cell types.

3. B cell follicle

Within the lymph node, B cells are mainly localized in follicles, which make up the outer cortex of the lymph node. Some follicles have a germinal center, where activated B cells are undergoing differentiation and maturation.

4. T cell zone

T cells are distributed in the paracortical areas surrounding B cell follicles, also referred to as the deep cortex or T cell zone. This is where T cells interact with antigen-presenting dendritic cells.

5. Medulla

The medulla contains blood vessels, sinuses and medullary cords that contain antibody-secreting plasma cells. Lymph drains into the medullary sinuses and flows into the efferent lymphatic vessel.
6. **Artery**

   The artery carries lymphocytes to the lymph node. The lymphocytes pass the capillary beds and continue to the high endothelial venules (HEV) from which they enter the lymph node.

7. **Vein**

   The vein collects the blood that supplies the lymph node and a small amount of lymph absorbed by the parenchymal blood vessels.

8. **Efferent lymphatic vessel**

   Lymph and lymphocytes leave the lymph node through efferent lymphatic vessels, which empty into the lymphatic ducts, and drain into one of the subclavian veins, near the junction of the internal jugular veins. Valves in the lymphatic vessels enforce the one-way flow of lymph.

9. **Reticular network**

   The reticular network is a supportive structure throughout the entire lymph node, composed of stromal cells and collagen fibrils. They provide a crossroad for the interaction of rare clonal precursors of recirculating T and B cells with their cognate antigens, whether delivered by the dendritic cells or as free antigens.
CITED REFERENCES


Murphy, K., & Weaver, C. (2017a). Antigen Recognition by B-cell and T-cell Receptors. In Janeway’s Immunology, 9th edition (p. 139).


Murphy, K., & Weaver, C. (2017c). Lymphocytes activated by antigen give rise to clones of antigen-specific effector cells that mediate adaptive immunity. In Janeway’s Immunology, 9th edition (p. 15).


70
GENERAL REFERENCES


Vita

Li Yao was born on October 22nd 1988, in Changsha city, Hunan province, China. In high school, Li joined a biological team to attend the International Biology Olympiad. Since then, her interest in biology has grown, and she decided to pursue a career related to life science. In 2007, Li entered Zhejiang University and studied Biological Science. She also received interdisciplinary training in Chu Kochen Honors College where she was inspired by many talented students and mentors. As an undergraduate student, Li had three years of experience doing research on understanding mechanisms of studying and long-term memory in a neuroscience lab in Zhejiang University School of Medicine.

To continue her career in science, Li began her graduate studies in the field of Genetics, Genomics and Development at Cornell University, Ithaca, NY, in 2011. While working on her PhD thesis on nuclear architecture and transcription regulation, she settled down and planned for her future career. Since Li always had a keen interest in improving communication in the science community and she enjoys drawing, she decided to continue her education in the Medical and Biological Illustration program in the Department of Art as Applied to Medicine at the Johns Hopkins University School of Medicine in 2015. While studying for her graduate degree, Li has received: an Award of Excellence from the Association of Medical Illustrators, the Frank H. Netter Memorial Scholarship for her academic performance at Johns Hopkins University, a Vesalian Scholar for her thesis work, and the Inez Demonet Scholarship from the Vesalius Trust. Two years of professional training in this new field has refreshed her understanding of medical and scientific communication and she hopes to contribute to the life sciences by pushing the standard of visual communication forward. Li received a Master of Science degree in May 2015. She will receive her Master’s degree in Medical and Biological Illustration in May 2017.