การออกแบบและพัฒนาชุดส่วนประกอบซอฟต์แวร์สำหรับสร้างภาพเชิงปริมาตร
เพื่อสนับสนุนงานด้านการแพทย์

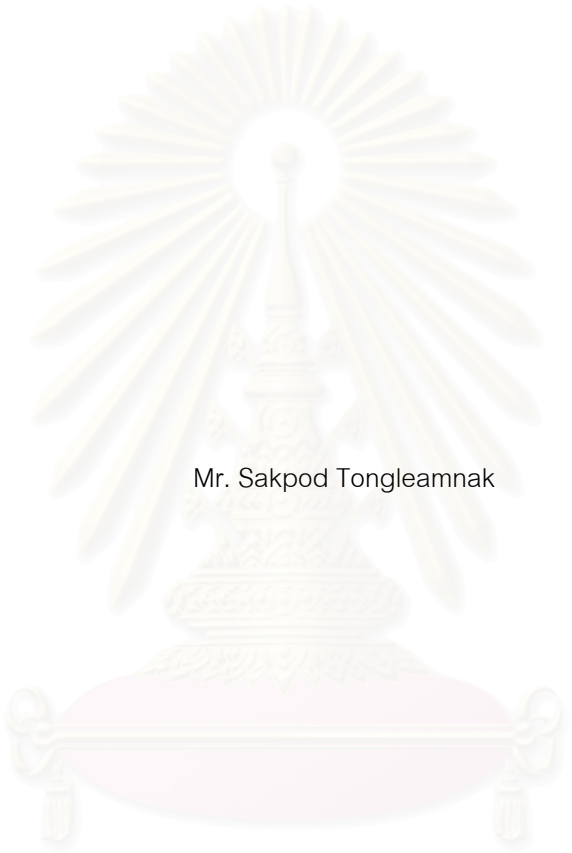
นายศักดิ์พจน์ ทองเลี่ยมนาค

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547

ISBN 974-17-6132-5

A DESIGN AND DEVELOPMENT OF SOFTWARE COMPONENTS FOR MEDICAL VOLUME VISUALIZATION

Mr. Sakpod Tongleamnak



A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2004

ISBN 974-17-6132-5

หัวข้อวิทยานิพนธ์	การออกแบบและพัฒนาชุดส่วนประกอบซอฟต์แวร์สำหรับสร้างภาพเชิงปริมาตรเพื่อสนับสนุนงานด้านการแพทย์
โดย	นายศักดิ์พจน์ ทองเลี่ยมนาค
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์นงลักษณ์ โควาวิสารัช
อาจารย์ที่ปรึกษาร่วม	ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดี คณะวิศวกรรมศาสตร์

(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการสอบ

(รองศาสตราจารย์ ดร.สมชาย ประสิทธิ์จูตระกูล)

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์นงลักษณ์ โควาวิสารัช)

..... อาจารย์ที่ปรึกษาร่วม

(ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... กรรมการ

(อาจารย์ ดร.ฐิต ศิริบูรณ์)

ศักดิ์พจน์ ทองเลี่ยมนาค: การออกแบบและพัฒนาชุดส่วนประกอบซอฟต์แวร์สำหรับสร้างภาพเชิงปริมาตรเพื่อสนับสนุนงานด้านการแพทย์ (A DESIGN AND DEVELOPMENT OF SOFTWARE COMPONENTS FOR MEDICAL VOLUME VISUALIZATION) อาจารย์ที่ปรึกษา: ผศ.นงลักษณ์ โควาวิสารัช, อาจารย์ที่ปรึกษาร่วม: ผศ.ดร.วิวัฒน์ วัฒนาวุฒิ, 91 หน้า. ISBN 974-17-6132-5.

ในปัจจุบันการสร้างภาพเชิงปริมาตรมีประโยชน์ทางการแพทย์อย่างมาก โดยเฉพาะการช่วยให้แพทย์เห็นสภาพภายในร่างกายของผู้ป่วยในรูปแบบของภาพสามมิติก่อนการวินิจฉัยหรือผ่าตัด แต่การพัฒนาโปรแกรมประยุกต์ในทางการแพทย์ที่มีการสร้างภาพเชิงปริมาตรต้องประสบปัญหายุ่งยากเนื่องจากความซับซ้อนและยุ่งยากของขั้นตอนวิธีในการสร้างภาพ นอกจากนี้ขั้นตอนวิธีส่วนใหญ่ยังให้ประสิทธิภาพในการสร้างภาพที่ต่ำ งานวิจัยนี้จึงมีวัตถุประสงค์เพื่อแก้ปัญหาเหล่านี้โดยได้ทำการออกแบบและพัฒนาชุดของส่วนโปรแกรมสำหรับสร้างภาพเชิงปริมาตรเพื่อใช้ในการสร้างโปรแกรมประยุกต์ทางด้านการแพทย์ที่มีการใช้การสร้างภาพเชิงปริมาตรเพื่อกำจัดความซับซ้อนและซ้ำซ้อนในการพัฒนาโปรแกรมประยุกต์

งานวิจัยนี้ได้ออกแบบและพัฒนาส่วนโปรแกรมสำหรับการสร้างภาพเชิงปริมาตรจำนวน 7 ส่วนโปรแกรม ได้แก่ VolumeX เป็นส่วนโปรแกรมหลักซึ่งใช้สร้างภาพเชิงปริมาตร TFFile เป็นส่วนโปรแกรมที่ใช้จัดเก็บและอ่านฟังก์ชันถ่ายโอน TFGraph เป็นส่วนโปรแกรมที่ใช้สร้างฟังก์ชันถ่ายโอนจากกราฟ TFGradient เป็นส่วนโปรแกรมที่ใช้สร้างฟังก์ชันถ่ายโอนจากการไล่โทนสี TFSelectColor เป็นส่วนโปรแกรมที่ใช้สร้างฟังก์ชันถ่ายโอนจากการเลือกสีของผู้ใช้ RawDataRead เป็นส่วนโปรแกรมสำหรับอ่านข้อมูลเชิงปริมาตร และ SWClsAndShading เป็นส่วนโปรแกรมสำหรับจำแนกประเภทและให้แสงเงาด้วยซอฟต์แวร์ โดยในส่วนโปรแกรม VolumeX งานวิจัยนี้ได้เลือกใช้ขั้นตอนวิธีในการสร้างภาพเชิงปริมาตรที่มีประสิทธิภาพสูงคือการสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์สามมิติซึ่งนำความสามารถของกราฟิกส์ฮาร์ดแวร์ในปัจจุบันมาช่วยเร่งความเร็วในการสร้างภาพเชิงปริมาตร รวมถึงได้เพิ่มความสามารถในการให้แสงเงา การจำแนกประเภท และความสามารถในการตัดส่วนของข้อมูลที่ไม่ต้องการออกทั้งโดยใช้ระนาบและโดยใช้รูปทรงเรขาคณิตโดยใช้กราฟิกส์ฮาร์ดแวร์เป็นตัวเร่งการทำงาน

จากการทดสอบในงานวิจัยพบว่าชุดส่วนโปรแกรมสามารถนำไปสร้างเป็นโปรแกรมประยุกต์สำหรับการสร้างภาพเชิงปริมาตรได้ และยังยืดหยุ่นเพียงพอกับกับการสร้างโปรแกรมประยุกต์บนเว็บเพจ ประสิทธิภาพโดยรวมของระบบอยู่ในเกณฑ์ดีแม้ว่าการให้แสงเงาจะส่งผลกระทบทำให้ประสิทธิภาพของการสร้างภาพต่ำลงแต่ก็ยังอยู่ในเกณฑ์ที่สามารถโต้ตอบกับผู้ใช้ได้

ภาควิชา	วิศวกรรมคอมพิวเตอร์	**ลายมือชื่อนิสิต**	.....
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์	**ลายมือชื่ออาจารย์ที่ปรึกษา**	.....
ปีการศึกษา	2547	**ลายมือชื่ออาจารย์ที่ปรึกษาร่วม**	.....

4470563021 : MAJOR COMPUTER SCIENCE
KEY WORD: VOLUME RENDERING, SOFTWARE COMPONENT, TEXTURE MAPPING, GRAPHICS HARDWARE, OPENGL

SAKPOD TONGLEAMNAK: A DESIGN AND DEVELOPMENT OF SOFTWARE COMPONENTS FOR MEDICAL VOLUME VISUALIZATION. THESIS ADVISOR: ASST. PROF. NONGLUK COVAVISARUCH THESIS CO-ADVISOR: ASST. PROF. WIWAT VATANAWOOD, Ph.D. 91 pp. ISBN 974-17-6132-5.

Nowadays volume rendering is very useful in medical area. A physician can visualize inside a patient's body in 3D before diagnosis or planning an operation. However, developing a medical application for volume visualization is complicated mainly because most volume rendering algorithms are complex. In addition, most visualization algorithms still have low performance. Therefore, the objectives of this thesis are to design and to develop a set of volume visualization software components for developing medical software applications. These software components are designed and developed in a way that should diminish the complications and redundancies in volume visualization software developing process.

Seven components are developed in this thesis as follows: VolumeX, the main component for volume rendering; TFFile, a component for saving and reading transfer function; TFGraph, a component for generating transfer function using graph; TFGradient, a component for generating transfer function using gradient; TFSelectColor, a component for generating transfer function using user's selected colors; RawDataRead, a component for reading volume data; and SWClsAndShading, a component for software classification and shading. In order to produce high performance visualization, the VolumeX component utilizes "Volume Rendering via 3D Texture Mapping" algorithm which uses modern graphics hardware capability to accelerate the rendering process. Hardware accelerated shading and clipping by a plane and a geometric volume are also developed as standard feature of the component.

The test results show that the components can be easily used to create a volume visualization application. It is also flexible enough to be used on a web-base application. Overall rendering performance is good. Even though shading can reduce some performance, rendering time is still low enough to instantly interact with a user.

Department	Computer Engineering	Student's signature	.....
Field of study	Computer Science	Advisor's signature	.....
Academic year	2004	Co-advisor's signature	.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงอย่างดี ก็ด้วยความช่วยเหลือความอนุเคราะห์และการสนับสนุนจากบุคคลหลาย ๆ ฝ่ายด้วยกัน ผู้วิจัยจะขอกล่าวต่อผู้มีพระคุณไว้ ณ ที่นี้

ขอขอบพระคุณคุณพ่อและคุณแม่ ผู้ซึ่งคอยให้การเลี้ยงดู อบรมสั่งสอน ให้กำลังใจและให้โอกาสในการศึกษาเล่าเรียนตลอดมา

ขอขอบพระคุณ ผศ.นงลักษณ์ โควาวิสารัช อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้ซึ่งคอยสละเวลาให้คำปรึกษา ความรู้ความเข้าใจ ข้อคิดเห็น โอกาส และความช่วยเหลือดูแลตลอดระยะเวลาในการศึกษาเป็นอย่างสูง

ขอขอบพระคุณ ผศ.ดร.วิวัฒน์ วัฒนาวุฒิ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ที่ได้ให้ความช่วยเหลือดูแลเอาใจใส่รวมทั้งให้คำปรึกษาและแนะนำสิ่งที่มีประโยชน์ตลอดการวิจัย

ขอขอบพระคุณภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ได้สนับสนุนเครื่องมือเครื่องใช้และสถานที่ในการทำวิจัย

ขอขอบคุณมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสารที่กรุณามอบทุนการศึกษาให้เป็นระยะเวลา 2 ปี

ขอขอบคุณ พี่สมนึก รังสีวงศ์ ที่ได้ช่วยเหลืออำนวยความสะดวกในการเดินทางและให้ข้อคิดเห็นต่าง ๆ ในงานวิจัย

ขอขอบคุณ Mr. George Zubal ที่กรุณาเอื้อเฟื้อชุดข้อมูลเชิงปริมาตรบางส่วน

ขอขอบคุณ คุณเจษฎา แสงพานิชย์ คุณเจษฎา ชินอานุภาพ คุณเฉลิมทรัพย์ คุณสราวุธิ คุณเอ็กซ์ พี่หนุ่ม พี่หรอย พี่แน็ท พี่หนุ่ย น้องกอล์ฟ น้องออย น้องอาร์ท น้องนัช น้องเพ็ญ น้องต้อง น้องโนนี่ ในมิตรภาพ ความมีน้ำใจ เอื้อเฟื้อเผื่อแผ่ ห่วงใยดูแลที่มีให้กันเสมอมา

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

รูปที่ หน้า

สารบัญภาพ (ต่อ)

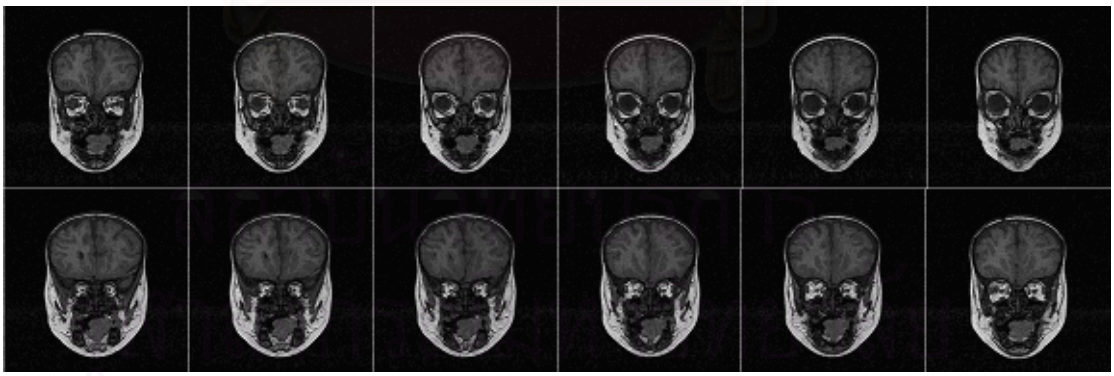
รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันเครื่องมือแพทย์จำนวนหนึ่งสามารถสร้างภาพตัดขวางของร่างกายผู้ป่วยได้ ยกตัวอย่างเช่นภาพที่ได้จากเครื่อง MRI (Magnetic Resonance Imaging) เครื่อง CT (Computed Tomography) ดังตัวอย่างเช่นในรูปที่ 1.1 ในการวินิจฉัยแต่ละครั้งแพทย์จะต้องตีความหมายและจินตนาการถึงลักษณะรูปร่างอวัยวะจริงใน 3 มิติ ซึ่งต้องใช้ภาพเป็นจำนวนมากเพื่อให้ครอบคลุมอวัยวะที่ต้องการจะตรวจ ในบางครั้งจะต้องใช้ภาพเป็นจำนวนหลายสิบหรืออาจจะถึงร้อยภาพ ทำให้การตีความชุดข้อมูลไปเป็นรูปร่างลักษณะใน 3 มิติทำได้ยากและอาจจะทำให้เกิดความผิดพลาดได้ การสร้างภาพเชิงปริมาตรจึงได้เข้ามามีบทบาทในการที่ช่วยให้แพทย์เห็นพยาธิสภาพภายในตัวผู้ป่วยในรูปแบบของภาพ 3 มิติได้เนื่องจากชุดภาพตัดขวางเหล่านี้สามารถแปลงเป็นข้อมูลเชิงปริมาตรได้ง่ายเพียงนำมาเรียงต่อกัน นอกจากจะเป็นประโยชน์ในการวินิจฉัยแล้วภาพสามมิติที่ได้จากการสร้างภาพเชิงปริมาตรยังสามารถประยุกต์ใช้ในงานด้านอื่นได้อีก เช่น ใช้วางแผนการผ่าตัด ใช้ประกอบคำอธิบายการรักษาแก่ผู้ป่วย ใช้ในด้านการศึกษา ฯลฯ



รูปที่ 1.1 ตัวอย่างข้อมูลด้านการแพทย์

ปัญหาที่สำคัญในการสร้างโปรแกรมประยุกต์ด้านการแพทย์ที่ต้องใช้การสร้างภาพเชิงปริมาตรคือความซับซ้อนของขั้นตอนวิธีในการสร้างภาพ ซึ่งส่งผลให้การพัฒนาโปรแกรมประยุกต์เป็นไปได้อย่างยากลำบาก เพื่อแก้ปัญหาความซับซ้อนและความซ้ำซ้อนในการสร้างโปรแกรมประยุกต์ทางด้านการแพทย์ที่จำเป็นต้องใช้การสร้างภาพเชิงปริมาตรงานวิจัยนี้จึงมีแนวคิดที่จะสร้างชุดส่วนโปรแกรม

สำหรับการสร้างภาพเชิงปริมาตรในด้านการแพทย์ขึ้น เพื่อให้ผู้พัฒนานำไปใช้ในการพัฒนาโปรแกรมประยุกต์ซึ่งสามารถช่วยลดความยุ่งยากในการพัฒนาไปได้มาก

นอกจากปัญหาความซับซ้อนแล้วขั้นตอนวิธีในการสร้างภาพเชิงปริมาตรส่วนมากที่มีมาในอดีตยังมีประสิทธิภาพในการสร้างภาพค่อนข้างต่ำการสร้างภาพเชิงปริมาตรในแต่ละภาพนั้นอาจจะใช้เวลาหลายสิบวินาทีจนถึงหลายนาทีขึ้นอยู่กับขนาดและความซับซ้อนของข้อมูลเชิงปริมาตร การสร้างภาพเชิงปริมาตรที่มีประสิทธิภาพสูงพอที่จะโต้ตอบกับผู้ใช้ได้โดยทันท่วงทีเกิดขึ้นบนฮาร์ดแวร์เฉพาะที่สร้างขึ้นเพื่อการสร้างภาพเชิงปริมาตรเท่านั้น ซึ่งฮาร์ดแวร์มีราคาสูงและไม่เป็นที่แพร่หลาย การขาดการโต้ตอบกับผู้ใช้อย่างทันท่วงทีทำให้การสร้างภาพเชิงปริมาตรเกิดความยุ่งยากและขาดความน่าสนใจในการใช้งานเป็นอย่างมาก เนื่องจากผู้ใช้ต้องรอเพื่อที่จะดูภาพผลลัพธ์ที่ได้จากการปรับเปลี่ยนพารามิเตอร์ในการสร้างภาพ เช่น การหมุน การสเกล เป็นต้น แต่ในปัจจุบันด้วยการพัฒนาอย่างรวดเร็วของกราฟิกส์ฮาร์ดแวร์ของเครื่องคอมพิวเตอร์ส่วนบุคคลทำให้เกิดขั้นตอนวิธีการสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์สามมิติ ซึ่งขั้นตอนวิธีนี้สามารถดึงความสามารถที่มีอยู่ในกราฟิกส์ฮาร์ดแวร์เหล่านี้มาใช้ในการเพิ่มประสิทธิภาพของการสร้างภาพเชิงปริมาตรจนกระทั่งอยู่ในระดับที่สามารถโต้ตอบกับผู้ใช้ได้อย่างทันท่วงที รวมถึงฮาร์ดแวร์เหล่านี้ยังมีราคาที่ไม่สูงนักและเป็นที่แพร่หลาย

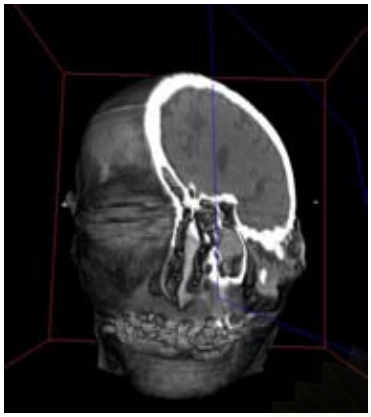
การให้แสงเงาเป็นความสามารถที่ควรให้ความสำคัญในการสร้างภาพ 3 มิติ เนื่องจากในการสร้างภาพ 3 มิตินั้นต้องทำการแสดงภาพของวัตถุ 3 มิติบนรูปที่เป็นระนาบ 2 มิติ การให้แสงเงาทำให้ภาพ 2 มิติที่ได้นั้นดูมีความลึกและมีมิติกว่ารูปที่ไม่มีการให้แสงเงา ดังแสดงในรูปที่ 1.2

(ก) รูปที่ได้จากการสร้างภาพเชิงปริมาตรที่ไม่มีการให้แสงเงา

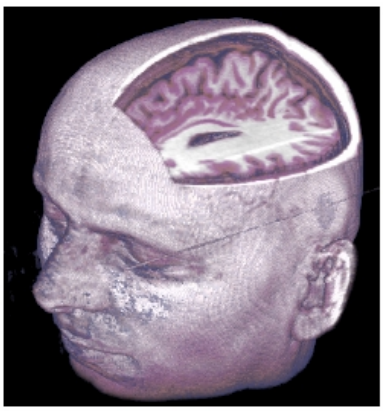
(ข) รูปที่ได้จากการสร้างภาพเชิงปริมาตรที่มีการให้แสงเงา

รูปที่ 1.2 เปรียบเทียบรูปจากการสร้างภาพเชิงปริมาตรที่มีการให้แสงเงาและไม่มีการให้แสงเงา

หนึ่งในข้อดีของข้อมูลเชิงปริมาตรคือสามารถเข้าไปดูภายในวัตถุได้ แต่การจะเข้าไปดูภายในวัตถุได้ต้องอาศัยการตัดเพื่อตัดเอาส่วนที่ไม่ต้องการด้านนอกออกให้เห็นส่วนที่อยู่ภายใน การตัดจึงเป็นความสามารถที่จำเป็นต้องมีสำหรับการสร้างภาพเชิงปริมาตร โดยทั่วไปแล้วการตัดจะกำหนดโดยใช้ระนาบและรูปทรง การตัดโดยใช้ระนาบคือการตัดเอาส่วนที่อยู่ฝั่งหนึ่งของระนาบใน 3 มิติออก การตัดโดยใช้รูปทรงสามารถกำหนดได้ว่าจะให้ตัดส่วนที่อยู่ภายในหรือตัดส่วนที่อยู่ภายนอกรูปทรงที่ใช้ตัดออก รูปที่ 1.3 แสดงการตัดโดยใช้ระนาบและการตัดโดยใช้รูปทรง



(ก) การตัดโดยใช้ระนาบ



(ข) การตัดโดยใช้รูปทรง

รูปที่ 1.3 การตัดในการสร้างภาพเชิงปริมาตร

กล่าวโดยย่อแล้วงานวิจัยนี้จะทำการออกแบบและพัฒนาชุดส่วนโปรแกรมสำหรับสร้างภาพเชิงปริมาตรเพื่อสนับสนุนงานด้านการแพทย์โดยชุดส่วนของโปรแกรมที่สร้างขึ้นจะใช้ขั้นตอนวิธีในการสร้างภาพที่มีประสิทธิภาพสูงพอที่จะโต้ตอบกับผู้ใช้ได้อย่างทันท่วงที สามารถให้แสงเงากับภาพที่สร้างขึ้น สนับสนุนการตัดผ่าทั้งโดยใช้ระนาบและรูปทรงเพื่อเข้าไปดูภายในข้อมูลเชิงปริมาตร และมีความยืดหยุ่นเพียงพอในการนำไปใช้กับโปรแกรมประยุกต์ที่หลากหลายรูปแบบและสภาพแวดล้อม

1.2 งานที่เกี่ยวข้อง

งานพัฒนาส่วนประกอบซอฟต์แวร์ที่ใช้ในการสร้างภาพเชิงปริมาตรที่มีมีก่อนหน้านี้ ได้แก่

1.2.1 VTK (Visualization Toolkit) โดย Kitware [1]

VTK เป็นส่วนประกอบซอฟต์แวร์แบบไลบรารีที่ใช้ในงานด้านการสร้างภาพทั่วไปโดยสามารถสร้างภาพได้ทั้งในเชิงพื้นผิว (surface rendering) และเชิงปริมาตร (volume rendering) อีกทั้งยังเปิดเผยรหัสต้นฉบับ (source code) ผู้ใช้สามารถนำไปใช้และแก้ไขได้โดยไม่เสียค่าใช้จ่าย และยังสามารถใช้ได้หลายแพลตฟอร์ม VTK สนับสนุนขั้นตอนวิธีการสร้างภาพเชิงปริมาตร 2 วิธีคือ วิธีการมองตามลำแสงและวิธี 2D Texture mapping hardware ไม่สนับสนุนการตัดโดยใช้รูปทรง และไม่สนับสนุนการใช้ฟังก์ชันถ่ายโอน (transfer function) 2 มิติ

1.2.2 Volumizer โดย SGI [2]

Volumizer เป็นส่วนประกอบซอฟต์แวร์แบบไลบรารีที่ทำหน้าที่เป็น API (Application Programming Interface) สำหรับสร้างภาพเชิงปริมาตร สร้างขึ้นบนโอเพ็นจีแอล (OpenGL) สามารถสร้างภาพเชิงปริมาตรจากชุดข้อมูลที่มีขนาดใหญ่มากได้ แต่มีข้อเสียคือสามารถทำงานได้เฉพาะบนเครื่องกราฟิกส์เวิร์คสเตชันระดับสูงเท่านั้น

1.2.3 Volumaid โดย Aicom Software [3]

Volumaid เป็นส่วนประกอบซอฟต์แวร์ทางการค้าแบบแอ็กทิฟเอกซ์คอนโทรล (ActiveX control) เพื่อสร้างภาพเชิงปริมาตร สร้างขึ้นบนโอเพ็นจีแอลข้อเสียของ Volumaid คือใช้การสร้างพื้นผิวจากข้อมูลที่เป็นเชิงปริมาตรโดยใช้ค่าขีดแบ่งจากนั้นจึงสร้างภาพจากพื้นผิวที่ได้โดยใช้การสร้างภาพเชิงพื้นผิว ทำให้ไม่สามารถกำหนดฟังก์ชันถ่ายโอนกับข้อมูลเชิงปริมาตรได้ และการตัดทำได้แค่เพียงการตัดโดยใช้ระนาบเท่านั้น

1.3 วัตถุประสงค์

เพื่อพัฒนาชุดส่วนประกอบซอฟต์แวร์สำหรับสร้างภาพเชิงปริมาตรจากเพื่อสนับสนุนงานด้านการแพทย์

1.4 ขอบเขตการวิจัย

ขอบเขตการทำงานของชุดแอ็กทิฟเอกซ์คอนโทรลที่สร้างขึ้นมีดังนี้

1) ในส่วนอ่านแฟ้มข้อมูลนำเข้าสามารถอ่านแฟ้มที่อยู่ในรูปแบบของแฟ้มข้อมูลดิบได้
2) สนับสนุนฟังก์ชันถ่ายโอน 1 หรือ 2 มิติ
3) การให้แสงเงาจะใช้เฉพาะแสงแวดล้อมและสะท้อนแบบกระจาย จากแหล่งกำเนิดแสงแหล่งเดียวที่เป็นจุดอยู่ที่อนันต์
4) การตัดผ่าวัตถุสามารถทำได้โดยระนาบ หรือรูปทรงพื้นฐานที่มีลักษณะปิด ได้แก่ รูปทรงสี่เหลี่ยม ทรงกลม หรือทรงกระบอก
5) มีฟังก์ชันพื้นฐานในการจัดการวัตถุ 3 มิติ ได้แก่ การเปลี่ยนขนาด (Scale) และการหมุน (Rotate)
6) ข้อมูลที่แต่ละแอ็กทิฟเอกซ์คอนโทรลในชุดใช้ร่วมกันหรือต้องส่งผ่านให้กันจะอยู่ในรูปแบบเดียวกัน

1.5 ขั้นตอนการดำเนินงานวิจัย

1) ศึกษาวิธีการในการสร้างภาพเชิงปริมาตร
2) ศึกษาโอเพ็นจีแอลและหน่วยประมวลผลภาพกราฟิกส์ของคอมพิวเตอร์ส่วนบุคคล
3) ทดลองสร้างภาพเชิงปริมาตรกับข้อมูลปริมาตรที่มีอยู่
4) ศึกษาแอ็กทิฟเอกซ์คอนโทรลและวิธีการสร้าง
5) ทดลองสร้างแอ็กทิฟเอกซ์คอนโทรลที่สามารถสร้างภาพเชิงปริมาตร
6) ออกแบบการติดต่อของแอ็กทิฟเอกซ์คอนโทรลกับคอนโทรลคอนเทนเนอร์
7) ศึกษาวิธีในการตัดผ่าวัตถุ
8) ทดลองสร้างภาพเชิงปริมาตรแล้วตัดผ่าวัตถุ

9) ศึกษาวิธีการให้แสงเงากับวัตถุในภาพ

10) ทดลองสร้างภาพเชิงปริมาตรแล้วให้แสงเงากับวัตถุ

11) สร้างแอ็กทิฟเอกซ์คอนโทรลตามที่ออกแบบไว้

12) ทดสอบการทำงานและวิเคราะห์ผล

13) สรุปและวิจารณ์ผลการทดลอง

14) จัดทำรายงาน

1.6 ประโยชน์ที่ได้รับ

ผลของงานวิจัยนี้จะได้ชุดส่วนประกอบซอฟต์แวร์สำหรับใช้สร้างภาพเชิงปริมาตรที่สนับสนุนงานด้านการแพทย์

1.7 โครงสร้างของวิทยานิพนธ์

วิทยานิพนธ์ฉบับนี้ได้กล่าวถึงความเป็นมาของปัญหา วัตถุประสงค์ ขอบเขต ขั้นตอนการวิจัย และประโยชน์ที่ได้รับจากงานวิจัยไว้แล้วในบทที่ 1 สำหรับบทที่ 2 จะได้กล่าวถึงหลักการและทฤษฎีที่เกี่ยวข้องกับงานวิจัยนี้ ในบทที่ 3 แสดงให้เห็นถึงภาพรวมของระบบและรายละเอียดการออกแบบส่วนโปรแกรมแต่ละส่วน รายละเอียดของขั้นตอนวิธีที่ใช้ในการสร้างภาพในงานวิจัยรวมถึงขั้นตอนวิธีการตัดผ่าจะถูกอธิบายในบทที่ 4 ผลการทดลอง สรุปและข้อเสนอแนะอยู่ในบทที่ 5 และ 6 ตามลำดับ

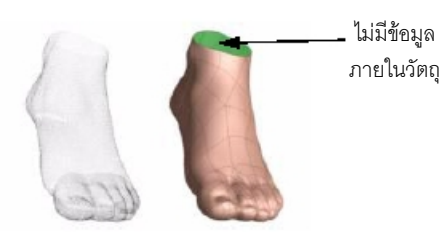
บทที่ 2

หลักการและทฤษฎีที่เกี่ยวข้อง

ทฤษฎีและหลักการที่ใช้ในงานวิจัยประกอบไปด้วย การสร้างภาพเชิงปริมาตรซึ่งอธิบายถึงทฤษฎีและกระบวนการที่ใช้ในการสร้างภาพจากข้อมูลเชิงปริมาตร กราฟิกส์ฮาร์ดแวร์จะอธิบายถึงคุณสมบัติขั้นตอนการทำงานของกราฟิกส์ฮาร์ดแวร์ การสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์สามมิติเป็นขั้นตอนวิธีในการสร้างภาพเชิงปริมาตรที่ใช้กราฟิกส์ฮาร์ดแวร์เข้ามาช่วยเร่งความเร็วซึ่งเป็นขั้นตอนวิธีหลักในการสร้างภาพเชิงปริมาตรของงานวิจัยนี้ การให้แสงเงาจะอธิบายถึงแบบจำลองการสะท้อนแสงของวัตถุที่ใช้ในการให้แสงเงาและวิธีการคำนวณการให้แสงเงา การหาเวกเตอร์ปกติของข้อมูลเชิงปริมาตรเป็นการอธิบายวิธีการที่ใช้ในการหาเวกเตอร์ปกติซึ่งเป็นสิ่งจำเป็นต้องใช้ในการให้แสงเงาจากข้อมูลเชิงปริมาตร แอ็กทิฟเอกซ์คอนโทรลจะอธิบายลักษณะและการใช้งานของส่วนโปรแกรมแบบแอ็กทิฟเอกซ์คอนโทรลซึ่งเป็นรูปแบบของส่วนโปรแกรมที่จะพัฒนา

2.1 การสร้างภาพเชิงปริมาตร

การสร้างภาพเชิงปริมาตรหมายถึงการสร้างภาพ 2 มิติจากชุดข้อมูลที่อยู่ในรูปปริมาตร 3 มิติ ซึ่งจะแตกต่างจากการสร้างภาพเชิงพื้นผิวที่ใช้ในงานคอมพิวเตอร์กราฟิกส์โดยทั่วไป เหตุผลที่การสร้างภาพเชิงปริมาตรไม่เป็นที่นิยมเท่ากับการสร้างภาพเชิงพื้นผิวนั้นเนื่องมาจากหลายสาเหตุ เริ่มจากปริมาณข้อมูลของวัตถุต่างๆ ในฉากที่ต้องเก็บในการสร้างภาพเชิงปริมาตรนั้นมีมากกว่าการสร้างภาพเชิงพื้นผิวซึ่งเก็บข้อมูลเฉพาะพื้นผิวภายนอกของวัตถุมาก เมื่อข้อมูลมีปริมาณมากขั้นตอนวิธีที่ใช้ในการสร้างภาพเชิงปริมาณจึงยุ่งยากมากกว่าการสร้างภาพเชิงพื้นผิวซึ่งส่งผลให้ประสิทธิภาพที่ได้ในการสร้างภาพต่ำกว่าการสร้างภาพเชิงพื้นผิว รวมถึงความจริงที่ว่างานประยุกต์ใช้คอมพิวเตอร์กราฟิกส์ส่วนใหญ่ไม่ได้สนใจภาพสิ่งที่อยู่ภายในของวัตถุเนื่องจากโดยปกติแล้วจะถูกพื้นผิวภายนอกของวัตถุบังจนหมด แต่การสร้างภาพเชิงปริมาตรกลับเป็นส่วนสำคัญในการประยุกต์จำนวนหนึ่งซึ่งข้อมูลภายในของวัตถุเป็นสิ่งที่มองข้ามไม่ได้ เช่น ในงานด้านการแพทย์ งานคำนวณด้านวิทยาศาสตร์หรือวิศวกรรมที่ให้ผลออกมาเป็นปริมาตร รวมถึงการสร้างภาพของวัตถุที่ไม่สามารถหาขอบเขตหรือพื้นผิวที่แน่นอนได้ เช่น หมอกหรือควัน รูปที่ 2.1 เป็นการเปรียบเทียบการสร้างภาพเชิงปริมาตรและการสร้างภาพเชิงพื้นผิว จะเห็นว่าการสร้างภาพเชิงพื้นผิวในรูปที่ 2.1 ก แม้ว่าทำให้พื้นผิวของเท้าใสสิ่งที่อยู่ภายในก็ว่างเปล่า ผิดกับการสร้างภาพเชิงปริมาตรในรูปที่ 2.1 ข ที่เมื่อทำให้ภายนอกใสจะสามารถเห็นสิ่งที่อยู่ภายในได้



(ก) การสร้างภาพเชิงพื้นผิว

(ข) การสร้างภาพเชิงปริมาตร

รูปที่ 2.1 เปรียบเทียบระหว่างการสร้างภาพเชิงปริมาตรและการสร้างภาพเชิงพื้นผิว

การสร้างภาพเชิงปริมาตรสามารถแบ่งได้ออกเป็น 2 ประเภทใหญ่ๆ คือ การสร้างภาพเชิงปริมาตรโดยตรง และการสร้างภาพเชิงปริมาตรโดยอ้อม การสร้างภาพเชิงปริมาตรโดยตรงจะสร้างภาพจากข้อมูลปริมาตรโดยไม่มีการเปลี่ยนแปลงรูปแบบของข้อมูลเชิงปริมาตร ในขณะที่การสร้างภาพเชิงปริมาตรโดยอ้อมจะทำการแปลงข้อมูลเชิงปริมาตรให้เป็นข้อมูลเชิงพื้นผิวแล้วใช้การสร้างภาพเชิงพื้นผิวในการสร้างภาพของข้อมูลที่ได้จากการแปลง การแปลงข้อมูลเชิงปริมาตรให้กลายเป็นข้อมูลเชิงพื้นผิวนั้นสามารถทำได้หลายวิธี วิธีที่นิยมใช้คือการกำหนดขอบเขตของวัตถุในข้อมูลเชิงปริมาตรโดยใช้ค่าขีดแบ่งกำหนดให้ตำแหน่งในปริมาตรที่มีค่าเท่ากับค่าขีดแบ่งนั้นเป็นขอบเขตของวัตถุแล้วใช้ขั้นตอนวิธีต่าง ๆ เช่น มาร์ชชิงคิวบ์[4] (Marching Cubes) ในการแปลงขอบเขตนั้นให้เป็นพื้นผิว นอกจากการสร้างภาพเชิงปริมาตรโดยอ้อมจะทำให้เกิดการสูญเสียข้อมูลภายในของวัตถุซึ่งเป็นข้อดีของการสร้างภาพเชิงปริมาตรในระหว่างการแปลงข้อมูลแล้ว ในกรณีที่กำหนดตำแหน่งขอบเขตของวัตถุไม่ละเอียดพอพื้นผิวที่ได้จะมีความขรุขระไม่ต่อเนื่องซึ่งจะส่งผลให้ภาพที่ได้มีคุณภาพไม่ดีเท่าที่ควร ภาพที่ 2.2 แสดงความแตกต่างระหว่างภาพที่ได้จากการสร้างภาพเชิงปริมาตรโดยตรงและการสร้างภาพเชิงปริมาตรโดยอ้อม เนื่องจากงานวิจัยนี้สนใจคุณสมบัติในการมองเห็นสิ่งที่อยู่ภายในวัตถุ การสร้างภาพเชิงปริมาตรโดยอ้อมจึงไม่อยู่ในความสนใจของงานวิจัย ดังนั้นการสร้างภาพเชิงปริมาตรที่จะกล่าวถึงต่อไปในวิทยานิพนธ์นี้จะหมายถึงเฉพาะการสร้างภาพเชิงปริมาตรโดยตรงเท่านั้น

รูปที่ 2.2 เปรียบเทียบภาพลำไส้ใหญ่ของมนุษย์ที่ได้จากการสร้างภาพเชิงโดยตรง (ซ้าย) และการสร้างภาพเชิงปริมาตรโดยอ้อม (ขวา)

ต่อไปจะกล่าวถึงทฤษฎีในการสร้างภาพเชิงปริมาตรซึ่งจะประกอบไปด้วย นิยามของข้อมูลเชิงปริมาตร การชักตัวอย่างและการประกอบให้คืนสภาพ แบบจำลองแสง การฉายแสง ฟังก์ชันถ่ายโอนและการแยกประเภท

2.1.1 **ข้อมูลเชิงปริมาตร** (Volume Data)

ตามนิยามข้อมูลเชิงปริมาตรคือสนามค่าสเกลาร์ในสามมิติ ซึ่งสามารถเขียนเป็นสมการทางคณิตศาสตร์ได้เป็น

$$f(\vec{x}) \in IR \text{ เมื่อ } \vec{x} \in IR^3 \qquad \ldots(2.1)$$

แม้ว่าโดยนิยามแล้วโดเมนของสนามปริมาตรจะมีลักษณะต่อเนื่องเป็นเซตของจำนวนจริงในสามมิติ (IR^3) แต่ในการสร้างภาพเชิงปริมาตรจะใช้ข้อมูลในลักษณะของปริมาตรที่ไม่ต่อเนื่องซึ่งได้มาจากการชักตัวอย่างจากปริมาตรต่อเนื่องในนิยาม ปริมาตรเล็กๆ ซึ่งใช้ค่าสเกลาร์แต่ละค่าที่ได้จากการชักตัวอย่างเป็นตัวแทนจะเรียกว่า จุดปริมาตร (Voxel - Volume Element) โดยทั่วไปค่าสเกลาร์ของจุดปริมาตรจะถูกเก็บอยู่ในรูปของแถวลำดับสามมิติ

2.1.2 **การชักตัวอย่างและการประกอบให้คืนสภาพ** (Sampling and Reconstruction)

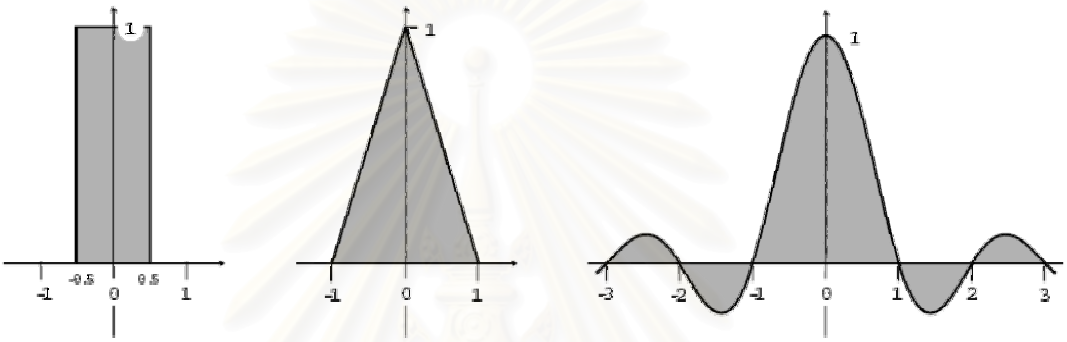
ในขั้นตอนสร้างภาพเชิงปริมาตรโดยทั่วไปแล้วจะมีการชักตัวอย่างข้อมูลเชิงปริมาตรใหม่อีกครั้งหนึ่ง โดยจุดที่ชักตัวอย่างใหม่อีกครั้งนี้ส่วนใหญ่แล้วจะไม่ตรงกับตำแหน่งเดิมที่เคยชักตัวอย่างจากปริมาตรต่อเนื่อง เพราะว่ามักจะมีการแปลงทั้งตำแหน่ง ขนาด และทิศทางของปริมาตรเพื่อให้ได้ภาพของข้อมูลเชิงปริมาตรในมุมมองและตำแหน่งต่างๆ ตามที่ผู้ใช้ต้องการ จึงต้องใช้การประกอบให้คืนสภาพเข้ามาช่วยในการคืนสภาพของปริมาตรต่อเนื่องขึ้นมาใหม่จากปริมาตรที่ไม่ต่อเนื่องเพื่อให้สามารถชักตัวอย่างใหม่ที่ตำแหน่งอื่นๆ ไม่ตรงกับจุดชักตัวอย่างเดิมได้

การประกอบให้คืนสภาพสามารถทำได้โดยการใช้ ตัวกรองการประกอบให้คืนสภาพ (Reconstruction filter) กับข้อมูลที่ไม่ต่อเนื่อง ตัวกรองการประกอบให้คืนสภาพที่ง่ายที่สุดคือ ตัวกรองแบบกล่อง (รูปที่ 2.3 ก) ซึ่งให้ผลเป็นการประมาณค่าในช่วงแบบเพื่อนบ้านที่ใกล้ที่สุด (Nearest Neighbor Interpolation) ตัวกรองแบบกล่องนี้ค่าที่ได้ในการชักตัวอย่างอีกครั้งจะเท่ากับค่า ณ ตำแหน่งชักตัวอย่างที่ใกล้ที่สุด ข้อดีของตัวกรองแบบนี้คือง่ายและมีประสิทธิภาพสูงแต่มีข้อเสียคือให้คุณภาพของการชักตัวอย่างแย่ที่สุดและก่อให้เกิดรอยหยักในข้อมูล ตัวกรองการประกอบให้คืนสภาพในรูป 2.3 ข คือตัวกรองการประกอบให้คืนสภาพแบบเต็นท์ ซึ่งให้ผลเป็นการประมาณค่าในช่วงแบบเชิงเส้น (Linear Interpolation) ตัวกรองนี้จะสมมติให้จุดชักตัวอย่างที่ล้อมรอบจุดชักตัวอย่างใหม่มีความสัมพันธ์แบบเชิงเส้น ค่าของจุดชักตัวอย่างใหม่จึงได้มาจากการแก้ความสัมพันธ์เชิงเส้นกับจุดชักตัวอย่างที่อยู่ล้อมรอบ ข้อดีของตัวกรองนี้คือค่อนข้างเร็วและคุณภาพของการสุ่มตัวอย่างดี ตัวกรองการประกอบให้คืนสภาพใน

รูปที่ 2.3 ค คือตัวกรองการประกอบให้คืนสภาพแบบซินซ์ (Sinc filter) ซึ่งถือว่าเป็นตัวกรองการประกอบให้คืนสภาพในอุดมคติ ตัวกรองนี้ใช้ ฟังก์ชันซินซ์ (Sinc function) ซึ่งสามารถเขียนใน โดเมนเชิงพื้นที่ (Spatial domain) ได้เป็น

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \qquad \text{...(2.2)}$$

ข้อเสียของตัวกรองแบบซินซ์นี้คือนำมาใช้ได้ยากในทางปฏิบัติเนื่องจากเป็นฟังก์ชันที่ไม่มีทางเท่ากับศูนย์ ไม่ว่าจะขยายโดเมนออกไปไกลแค่ไหน ทั้งยังใช้เวลาในการคำนวณเป็นเวลานาน



(ก) ตัวกรองแบบกล่อง *(ข) ตัวกรองแบบเต็นท์* *(ค) ตัวกรองแบบซินซ์*

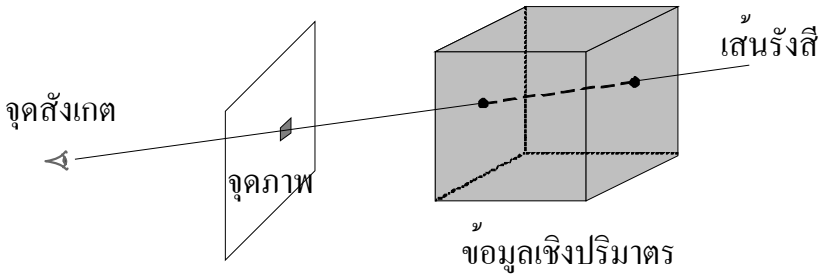
รูปที่ 2.3 ตัวกรองการประกอบให้คืนสภาพ

2.1.3 **แบบจำลองแสง** (Optical Model)

แบบจำลองแสงเป็นสิ่งจำเป็นในการสร้างภาพเชิงปริมาตร เนื่องจากเป็นสิ่งที่ใช้ในการกำหนดคุณสมบัติแสงให้แก่ข้อมูลเชิงปริมาตรเพื่อใช้ในการสร้างภาพ แบบจำลองแสงนั้นมีหลายแบบจำลองแต่แบบจำลองที่นิยมใช้ในการสร้างภาพเชิงปริมาตรได้แก่ แบบการดูดกลืนแสงรวมกับการเปล่งแสง (Absorption plus emission) ซึ่งกำหนดให้แต่ละอนุภาคภายในข้อมูลเชิงปริมาตรมีการเปล่งแสง และสามารถดูดกลืนแสงที่ผ่านเข้ามาได้ แต่จะไม่มีการกระเจิงของแสง

2.1.4 **การฉายแสง** (Ray-Casting)[5]

การฉายแสงเป็นวิธีที่ใช้ในการสร้างภาพเชิงปริมาตรโดยเริ่มจากจุดสังเกตสมมติลากเส้นรังสีผ่านจุดศูนย์กลางของจุดภาพที่ต้องการจะหาค่า ดังรูปที่ 2.4 จากแบบจำลองแสงที่ใช้คือแบบการดูดกลืนแสงรวมกับการเปล่งแสง ค่าสีที่อยู่ที่จุดภาพจะเท่ากับค่าสีของแสงที่อนุภาคภายในข้อมูลเชิงปริมาตรที่อยู่ตลอดตามแนวเส้นรังสีเปล่งและดูดกลืน ซึ่งสามารถเขียนสมการปริพันธ์ของการการหาค่าสีนี้ได้ดังสมการที่ 2.3 เมื่อทำการหาค่าสีได้จนครบทุกจุดภาพก็จะได้ภาพของข้อมูลเชิงปริมาตรนั้น



รูปที่ 2.4 การสร้างภาพเชิงปริมาตรโดยวิธีการฉายแสง

$$C = \int_0^D c(s(\vec{x}(t)))e^{-\int_0^t \tau(s(\vec{x}(t')))dt'} dt \qquad \text{...(2.3)}$$

เมื่อ $\vec{x}(t)$ คือแนวรังสีที่ฉายเข้าไปยังปริมาตรโดย t คือระยะทางจากจุดที่มอง $s(\vec{x}(t))$ คือค่าสเกลาร์ของปริมาตร ณ ตำแหน่งนั้น C คือค่าสีผลลัพธ์ของจุดภาพ ซึ่งหาได้จากการหาปริพันธ์ของค่าสี $c(s(\vec{x}(t)))$ สำหรับการเปล่งแสง และสัมประสิทธิ์การดูดกลืนแสง $\tau(s(\vec{x}(t)))$ ตลอดแนวเส้นรังสีจนถึงระยะทาง D

แต่ในทางปฏิบัติแล้วไม่สามารถทำการประเมินค่าปริพันธ์ในสมการที่ 2.3 ได้โดยตรง เนื่องจากตัวข้อมูลเชิงปริมาตรเป็นข้อมูลที่อยู่ในลักษณะไม่ต่อเนื่อง ดังนั้นจึงต้องใช้การประมาณค่าและการหาปริพันธ์เชิงตัวเลขเพื่อเปลี่ยนสมการแบบปริพันธ์เป็นสมการแบบรวมยอด

$$e^{-\int_0^t \tau(s(\vec{x}(t')))dt'} \approx e^{-\sum_{i=0}^{\lfloor t/d \rfloor} \tau(s(\vec{x}(id)))d} \qquad \text{...(2.4)}$$

สมการที่ 2.4 เป็นการประมาณค่าเชิงตัวเลขของพจน์การดูดกลืนแสงถึงตำแหน่ง $\vec{x}(t)$ ในสมการที่ 2.3 เมื่อ d คือระยะห่างระหว่างจุดสุ่มตัวอย่าง

$$e^{-\sum_{i=0}^{\lfloor t/d \rfloor} \tau(s(\vec{x}(id)))d} = \prod_{i=0}^{\lfloor t/d \rfloor} e^{-\tau(s(\vec{x}(id)))d} \qquad \text{...(2.5)}$$

สมการที่ 2.5 เป็นแทนที่การรวมยอดบนเลขชี้กำลังของสมการที่ 2.4 ด้วยการคูณของพจน์ยกกำลัง เมื่อกำหนดให้ A เป็นค่าการดูดกลืนแสงของวัตถุ โดยให้ $A_i = 1 - e^{-\tau(s(\vec{x}(id)))d}$ แล้วนำไปแทนในสมการที่ 2.5 จะได้สมการที่ 2.6

$$\prod_{i=0}^{\lfloor t/d \rfloor} e^{-\tau(s(\vec{x}(id)))d} = \prod_{i=0}^{\lfloor t/d \rfloor} (1 - A_i) \qquad \text{...(2.6)}$$

สมการที่ 2.6 ทำให้ใช้ A_i แทนค่าประมาณของการดูดกลืนแสง ณ ตำแหน่งที่ i บนแนวเส้นรังสีเพื่อหลีกเลี่ยงที่จะต้องคำนวณค่าของการดูดกลืนแสงทุก ๆ จุดบนรังสี ในทำนองเดียวกันพจน์ของการ

เปล่งแสง ณ ตำแหน่งที่ i ก็สามารถประมาณค่าได้ดังสมการที่ 2.7

$$C_i = c(s(\vec{x}(id)))d \quad \ldots(2.7)$$

จากสมการที่ 2.6 และ 2.7 เราสามารถเขียนสมการที่ 2.3 ใหม่ได้ดังสมการที่ 2.8 โดยให้ n คือจำนวนของจุดชักตัวอย่าง $\left(n = \lfloor D/d \rfloor\right)$

$$C_{approx} = \sum_{i=0}^{n} C_i \prod_{j=0}^{i-1}(1 - A_i) \quad \ldots(2.8)$$

จะสังเกตได้ว่าจุดชักตัวอย่างของสมการที่ 2.8 เป็นจุดชักตัวอย่างที่อยู่แนวเส้นรังสีซึ่งไม่ตรงกับตำแหน่งของจุดชักตัวอย่างเดิมของข้อมูลเชิงปริมาตร ดังนั้นจึงต้องใช้การประกอบให้คืนสภาพเข้ามาช่วยในการชักตัวอย่างบนเส้นรังสีนี้ และในการหาค่าสีจากสมการที่ 2.8 ต้องใช้ค่าสีและการดูดกลืนแสง (C_i, A_i) แต่จุดปริมาตรเป็นเพียงค่าสเกลาร์ดังนั้นจึงต้องมีฟังก์ชันสำหรับแปลงจากค่าสเกลาร์เป็นค่าสีและการดูดกลืนแสง ซึ่งฟังก์ชันนี้เรียกว่า ฟังก์ชันถ่ายโอน (transfer function) ซึ่งจะกล่าวถึงในหัวข้อถัดไป

$$C'_i = C_i + (1 - A_i)C'_{i+1} \quad \ldots(2.9)$$

สมการที่ 2.9 ได้มาจากการหาค่าของสมการที่ 2.8 จากหลังไปหน้า โดยลดค่า i จาก $n-1$ ถึง 0 ค่าสีใหม่ C'_i ได้มาจากค่าสีและค่าการดูดกลืนแสง (C_i, A_i) ที่ตำแหน่ง i และค่าสีที่ได้จากตำแหน่งก่อนหน้า C'_{i+1} โดยให้ค่าสีเริ่มต้น $C'_n = 0$

$$C'_i = C'_{i-1} + (1 - A'_{i-1})C_i \quad \ldots(2.10)$$

$$A'_i = A'_{i-1} + (1 - A'_{i-1})A_i \quad \ldots(2.11)$$

สมการที่ 2.10 และ 2.11 ได้มาจากการหาค่าของสมการที่ 2.8 จากหน้าไปหลัง โดยเพิ่มค่า i จาก 1 ถึง n ค่าสีและค่าการดูดกลืนแสงใหม่ (C'_i, A'_i) ได้มาจากค่าสี ค่าการดูดกลืนแสงที่ตำแหน่งนั้น (C_i, A_i) และค่าสี ค่าการดูดกลืนแสงที่ได้จากตำแหน่งก่อนหน้า (C'_{i-1}, C'_{i-1}) โดยค่าเริ่มต้นของ $C'_0 = 0$ และ $C'_0 = 0$

จะสังเกตได้ว่าการรวมค่าสีทั้งในสมการที่ 2.9 2.10 และ 2.11 ใช้ค่าการดูดกลืนแสงหรือค่าอัลฟาเป็นตัวกำหนดน้ำหนักของสีในการรวม ซึ่งการทำแบบนี้จะเรียกว่าอัลฟาเบล็นดิง (alpha blending) ความแตกต่างของการรวมจากหลังไปหน้าโดยใช้สมการที่ 2.9 กับหน้าไปหลังโดยใช้สมการ 2.10 และ 2.11 คือ การรวมสีจากหน้าไปหลังต้องมีการคำนวณการรวมค่าอัลฟาด้วยในขณะที่การรวมแบบหลังไปหน้าไม่ต้อง แต่ข้อดีของการรวมจากหน้าไปหลังคือเมื่อค่าอัลฟาที่ได้จากการรวมเท่ากับ 1

แล้วไม่ว่าค่าสีและค่าการดูดกลืนแสงที่ตำแหน่งถัดไปจะเป็นเท่าไรค่าสีที่ได้จะไม่มีการเปลี่ยนแปลงทำให้สามารถหยุดการคำนวณบนแนวเส้นรังสีนั้นได้โดยไม่ต้องคำนวณตลอดทั้งแนวเส้นรังสี

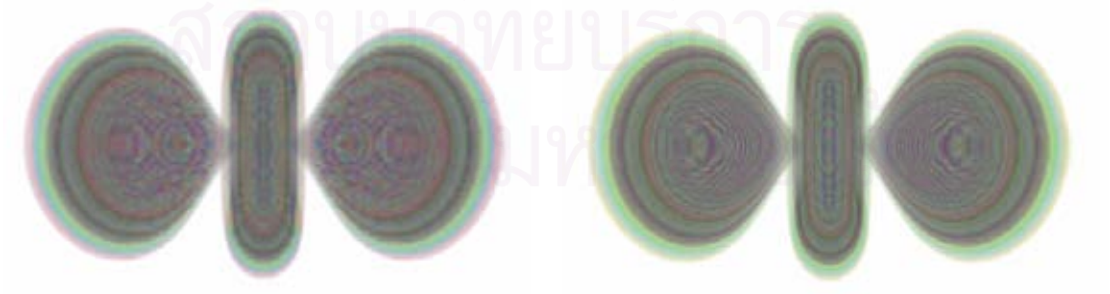
2.1.5 **ฟังก์ชันถ่ายโอนและการจำแนกประเภท** (Transfer Function and Classification)

ฟังก์ชันถ่ายโอนมีความสำคัญในการสร้างภาพเชิงปริมาตรเป็นอย่างมากเนื่องจากเป็นฟังก์ชันที่ใช้แปลงจากค่าสเกลาร์ของข้อมูลเชิงปริมาตรเดิมให้เป็นคุณสมบัติทางแสงของข้อมูลเชิงปริมาตรเพื่อใช้ในการสร้างภาพ ภาพของข้อมูลที่ได้จะมีลักษณะเป็นอย่างไรนั้นจะขึ้นอยู่กับฟังก์ชันถ่ายโอนซึ่งโดยทั่วไปแล้วจะกำหนดฟังก์ชันถ่ายโอนให้ค่าสเกลาร์ของที่สนใจมีคุณสมบัติทางแสงแตกต่างจากค่าสเกลาร์อื่น ๆ สมการที่ 2.9 คือสมการทั่วไปของฟังก์ชันถ่ายโอนของแบบจำลองแสงแบบการดูดกลืนแสงรวมกับการเปล่งแสง

$$\{C, A\} = T(s) \qquad \text{...(2.12)}$$

เมื่อ T คือฟังก์ชันถ่ายโอน และ s คือค่าสเกลาร์

การจำแนกประเภท (classification) คือการประยุกต์ฟังก์ชันถ่ายโอนโดยทั่วไปมี 2 วิธีคือ การจำแนกประเภทแบบก่อน (pre-classification) และการจำแนกประเภทแบบหลัง (post-classification) ทั้งสองวิธีมีความแตกต่างกันที่ลำดับในการประยุกต์ฟังก์ชันถ่ายโอน การจำแนกประเภทแบบก่อนจะทำการประยุกต์ฟังก์ชันถ่ายโอนโดยใช้สมการที่ 2.12 กับข้อมูลเชิงปริมาตรก่อนเพื่อแปลงจุดปริมาตรจากค่าสเกลาร์ไปเป็นคุณสมบัติทางแสง จากนั้นในการชักตัวอย่างและการประกอบให้คืนสภาพก็จะเป็นการประมาณค่าของคุณสมบัติทางแสง ในขณะที่การจำแนกประเภทแบบหลังจะทำการประยุกต์ฟังก์ชันถ่ายโอนหลังจากขั้นตอนการชักตัวอย่างและการประกอบให้คืนสภาพ ซึ่งค่าประมาณสเกลาร์ที่ได้จะถูกนำไปแทนในสมการที่ 2.12 เพื่อให้ได้ค่าสีและการดูดกลืนแสงที่ตำแหน่งนั้น ภาพที่ได้จากการจำแนกประเภททั้งสองวิธีนี้จะแตกต่างกันอย่างเห็นได้ชัดในกรณีที่ฟังก์ชันถ่ายโอนที่ใช้มีความถี่สูง ดังรูปที่ 2.5



(ก) *(ข)*

รูปที่ 2.5 (ก) ภาพที่ได้จากการจำแนกประเภทแบบก่อน (ข) ภาพที่ได้จากการจำแนกประเภทแบบหลัง

2.2 กราฟิกส์ฮาร์ดแวร์ (Graphics Hardware)

กราฟิกส์ฮาร์ดแวร์ที่จะกล่าวถึงในงานวิจัยนี้จะหมายถึงกราฟิกส์ฮาร์ดแวร์ของเครื่องคอมพิวเตอร์ส่วนบุคคลที่อยู่ในระดับของผู้บริโภคทั่วไป ซึ่งในปัจจุบันกราฟิกส์ฮาร์ดแวร์ในระดับนี้สามารถพบเห็นได้ทั่วไปและถูกติดตั้งอยู่ในเครื่องคอมพิวเตอร์ส่วนบุคคลส่วนใหญ่ หน้าที่หลักของกราฟิกส์ฮาร์ดแวร์คือรับคำสั่งจากหน่วยประมวลผลหลักเพื่อแสดงผลในลักษณะที่เป็นภาพกราฟิกส์ทั้ง 2 มิติและ 3 มิติ ซึ่งในงานวิจัยนี้จะกล่าวถึงเฉพาะการแสดงผลภาพกราฟิกส์ 3 มิติเท่านั้น

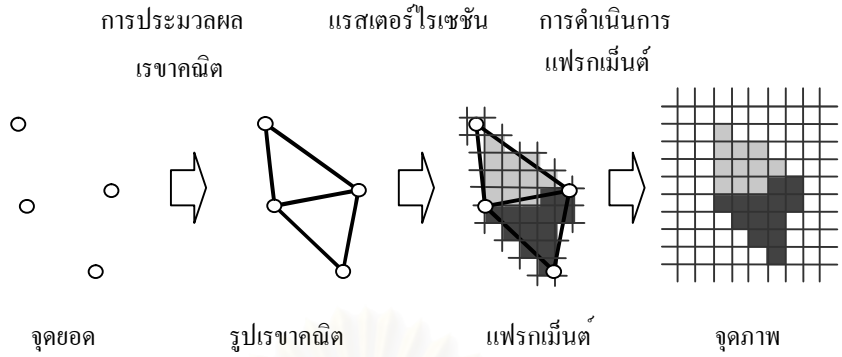
2.2.1 โอเพ็นจีแอล (OpenGL) [6]

โอเพ็นจีแอลคือส่วนต่อประสานโปรแกรมประยุกต์ (API-Application Program Interface) ที่ใช้ที่ใช้ในงานคอมพิวเตอร์กราฟิกส์ เพื่อใช้เป็นช่องทางให้โปรแกรมเมอร์ใช้ติดต่อกับกราฟิกส์ฮาร์ดแวร์ โอเพ็นจีแอลถูกออกแบบมาไม่ให้ขึ้นกับฮาร์ดแวร์หรือระบบปฏิบัติการ เพื่อต้องการให้มีประสิทธิภาพสูง โอเพ็นจีแอลจึงถูกสร้างให้อยู่ในระดับต่ำเพื่อให้สามารถติดต่อกับฮาร์ดแวร์ได้อย่างรวดเร็ว ดังนั้นจะไม่มีคำสั่งสำหรับสร้างรูปทรงต่าง ๆ เช่น ทรงกลม ลูกบาศก์ หรือสร้างความสัมพันธ์ระหว่างวัตถุต่าง ๆ ดังนั้นในการใช้โอเพ็นจีแอลผู้ใช้จำเป็นต้องสร้างวัตถุจากสิ่งพื้นฐาน ได้แก่ จุด เส้น และรูปหลายเหลี่ยม (polygon)

ด้วยคุณสมบัติของการเป็นส่วนต่อประสานโปรแกรมประยุกต์ทำให้โอเพ็นจีแอลกลายเป็นมาตรฐานสำหรับทั้ง ผู้พัฒนาโปรแกรมแสดงผลภาพกราฟิกส์ในการเขียนโปรแกรมเรียกใช้กราฟิกส์ฮาร์ดแวร์ และผู้ผลิตกราฟิกส์ฮาร์ดแวร์ในการกำหนดคุณลักษณะของตัวฮาร์ดแวร์ให้เข้ากันได้กับโอเพ็นจีแอล ซึ่งผู้ผลิตกราฟิกส์ฮาร์ดแวร์ของเครื่องคอมพิวเตอร์ส่วนบุคคลส่วนใหญ่ก็ใช้โอเพ็นจีแอลเป็นมาตรฐานหนึ่งในการกำหนดคุณลักษณะของกราฟิกส์ฮาร์ดแวร์

2.2.2 ขั้นตอนการแสดงผลภาพกราฟิกส์ 3 มิติ

กราฟิกส์ฮาร์ดแวร์จะทำการประมวลผลเพื่อแสดงผลภาพกราฟิกส์ 3 มิติโดยใช้การทำงานแบบสายท่อ (pipeline) โดยมีลำดับของขั้นตอนการทำงานที่ตายตัว ข้อมูลนำเข้าที่กราฟิกส์ฮาร์ดแวร์รับมาประมวลผลจะอยู่ในลักษณะของกระแสข้อมูลของจุดยอด (stream of vertices) ซึ่งจุดอยู่เหล่านี้สามารถนำมาต่อรวมกับเพื่อสร้างรูปร่างพื้นฐาน เช่น เส้นตรง สามเหลี่ยม และรูปหลายเหลี่ยม ข้อมูลส่งออกของขั้นตอนการงานคือภาพแรสเตอร์ (raster image) ของฉากเสมือนที่สร้างขึ้นซึ่งสามารถนำไปแสดงผลบนจอภาพได้ ขั้นตอนการทำงานภายในสายท่อสามารถแบ่งออกได้เป็น 3 ขั้นตอนหลัก ดังรูปที่ 2.6



รูปที่ 2.6 ขั้นตอนการแสดงผลภาพกราฟิกส์ 3 มิติ

1) **การประมวลผลเรขาคณิต** (Geometry Processing)

ในขั้นตอนนี้กระแสของจุดยอดที่เข้ามาจะถูกประมวลผลแบบทีละจุดยอด ซึ่งจะประกอบไปด้วย

i. การแปลงพิกัดของจุดยอดให้อยู่ในตำแหน่งที่ต้องการ โดยใช้การเลื่อน การหมุน และการสเกล
ii. การคำนวณการให้แสง โดยใช้เวกเตอร์ปกติของแต่ละจุดยอดร่วมกับทิศทางของแสงและของผู้สังเกต
iii. เชื่อมต่อจุดยอดต่าง ๆ เข้าด้วยกันเพื่อประกอบกันเป็นรูปเรขาคณิตพื้นฐาน
iv. ตัดส่วนของรูปเรขาคณิตพื้นฐานที่ไม่อยู่ภายในพื้นที่แสงผลทิ้งไป
v. ทำการแปลงพิกัดของจุดยอดต่าง ให้อยู่บนระนาบของภาพที่ต้องการสร้าง

ในขั้นตอนนี้มีการทำงานหนึ่งที่งานวิจัยนี้นำมาใช้ในการตัดโดยใช้รูปทรงจึงต้องกล่าวถึงรายละเอียดคือ การทดสอบคัลแฟซ (cull face) ซึ่งถูกกระทำในขณะที่กราฟิกส์ฮาร์ดแวร์แปลงค่าพิกัดจุดยอดของรูปทรงหลายเหลี่ยมในสามมิติให้อยู่ในระบบพิกัดของจอภาพ (v) โดยปกติแล้วกราฟิกส์ฮาร์ดแวร์จะกำหนดให้รูปหลายเหลี่ยมที่มีจุดยอดเรียงทวนเข็มนาฬิกาเป็นรูปหลายเลี่ยมที่หันด้านหน้าเข้าหาจอภาพ และในทางตรงข้ามรูปหลายเหลี่ยมที่มีจุดยอดเรียงตามเข็มนาฬิกาเป็นรูปหลายเลี่ยมที่หันด้านหลังเข้าหาจอภาพ แต่ผู้ใช้ก็สามารถกำหนดให้กราฟิกส์ฮาร์ดแวร์กำหนดให้รูปหลายเหลี่ยมที่มีจุดยอดเรียงตามเข็มนาฬิกาเป็นรูปหลายเลี่ยมที่หันด้านหน้าเข้าหาจอภาพได้เช่นกัน การทดสอบคัลแฟซเป็นการทดสอบที่ผู้ใช้สามารถกำหนดได้ว่าจะให้กราฟิกส์ฮาร์ดแวร์ตัดรูปหลายเหลี่ยมที่หันด้านใดออกไป โดยกำหนดได้สามค่าคือ ด้านหน้า ด้านหลัง และทั้งสองด้าน

2) **แรสเตอร์ไรเซชัน** (Rasterization)

ในขั้นตอนนี้กราฟิกส์ฮาร์ดแวร์จะแตกรูปเรขาคณิตพื้นฐานออกเป็นส่วนย่อย ๆ เรียกว่า แฟรกเม็นต์ (fragment) ซึ่งแฟรกเม็นต์แต่ละอันจะกลายเป็นจุดภาพแต่ละจุดของภาพผลลัพธ์ต่อไป และมีการทำเท็กซ์เจอร์แม็พพิง (texture mapping) ขั้นตอนแรสเตอร์ไรเซชันประกอบไปด้วยขั้นตอนย่อย ได้แก่

vi. การแตกรูปเรขาคณิตพื้นฐานออกเป็นแฟรกเม็นต์ โดยในการแตกนี้ค่าสี ค่าความสว่างที่ได้จากการให้แสงเงา และค่าพิกัดเท็กซ์เจอร์ของแฟรกเม็นต์จะได้มาจากการประมาณค่าในช่วงจากจุดยอด

vii. การอ่านค่าเท็กซ์เจอร์โดยใช้ค่าพิกัดเท็กซ์เจอร์ของแฟรกเม็นต์เป็นค่าบอกตำแหน่งที่อ่าน

viii. การทำแฟรกเม็นต์เชดดิง (fragment shading) เป็นการรวมค่าสีของแฟรกเม็นต์ที่ได้ทั้งจากการประมาณค่าในช่วง ค่าความสว่าง และค่าสีที่ได้จากการอ่านเท็กซ์เจอร์เข้าด้วยกันเพื่อให้เป็นค่าสีสุดท้ายของแฟรกเม็นซ์ซึ่งจะใช้เขียนลงบนเฟรมบัฟเฟอร์ (frame buffer) ถัดไป

รายละเอียดในการทำเท็กซ์เจอร์แม็พพิงและในหัวข้อที่ vii และ viii ซึ่งเป็นส่วนสำคัญในงานวิจัยนี้จะกล่าวถึงในหัวข้อ 2.2.3 และ 2.2.4

3) การดำเนินการแฟรกเม็นต์ (Fragment Operations)

ในขั้นตอนนี้แฟรกเม็นต์จะถูกทดสอบคุณสมบัติก่อนที่จะถูกเขียนลงบนเฟรมบัฟเฟอร์ ซึ่งถ้าแฟรกเม็นต์มีคุณสมบัติไม่ตรงตามเงื่อนไขที่กำหนดก็จะถูกตัดทิ้งโดยไม่มีการเขียนค่าแฟรกเม็นต์นั้นลงบนเฟรมบัฟเฟอร์ซึ่งจะประกอบไปด้วย

ix. การทดสอบค่าอัลฟา (alpha test) เป็นการทดสอบค่าอัลฟาของแฟรกเม็นต์ว่ามีคุณสมบัติตรงตามเงื่อนไขที่กำหนดหรือไม่

x. การทดสอบสเต็นซิล (stencil test) เป็นการทดสอบว่าที่ตำแหน่งของแฟรกเม็นต์นั้นค่าบนสเต็นซิลบัฟเฟอร์ (stencil buffer) ตรงตามเงื่อนไขที่กำหนดหรือไม่

xi. การทดสอบความลึก (depth test) เป็นการเปรียบเทียบค่าความลึกของแฟรกเม็นต์กับค่าที่อยู่บนบัฟเฟอร์ความลึก (depth buffer) ว่าตรงตามเงื่อนไขหรือไม่

xii. การทำอัลฟาเบล็นดิง (alpha blending) เมื่อแฟรกเม็นต์ผ่านการทดสอบทั้งหมดแล้วค่าสีของแฟรกเม็นต์จะถูกรวมกับค่าสีที่อยู่บนเฟรมบัฟเฟอร์โดยใช้ค่าอัลฟาเป็นตัวกำหนดน้ำหนักของค่าสี

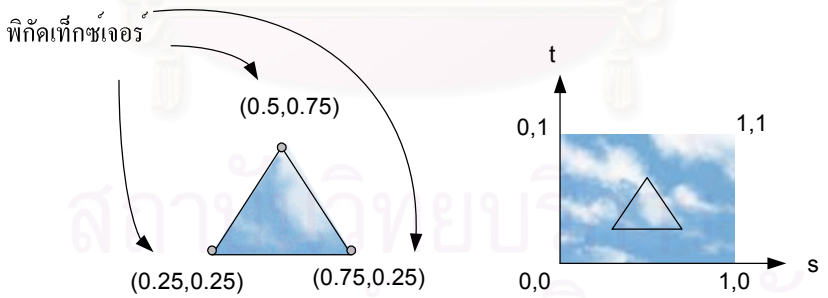
งานวิจัยนี้ใช้การทดสอบสเต็นซิลบัฟเฟอร์ในการตัดโดยใช้รูปทรงดังนั้นจึงต้องกล่าวถึงรายละเอียดของการทดสอบสเต็นซิลบัฟเฟอร์ สเต็นซิลบัฟเฟอร์คือบัฟเฟอร์ภายในกราฟิกส์ฮาร์ดแวร์ที่มีขนาดเท่ากับเฟรมบัฟเฟอร์แต่ใช้เก็บค่าตัวเลขแทนค่าสี ผู้ใช้ไม่สามารถเขียนหรืออ่านค่าใน สเต็นซิลบัฟเฟอร์ได้โดยตรง การใช้งานสเต็นซิลบัฟเฟอร์มี 2 องค์ประกอบคือ สเต็นซิลฟังก์ชัน (stencil function) และ การดำเนินการสเต็นซิล (stencil operation) สเต็นซิลฟังก์ชันมีหน้าที่กำหนดฟังก์ชันที่ใช้ในการทดสอบสเต็นซิล ผู้ใช้สามารถกำหนด ฟังก์ชัน ค่าอ้างอิง และมาสก์ (mask) โดยฟังก์ชันที่กำหนดได้ ได้แก่ ไม่ผ่านตลอด ผ่านตลอด น้อยกว่า น้อยกว่าหรือเท่ากับ เท่ากับ มากกว่าหรือเท่ากับ มากกว่า และไม่เท่ากับ แฟรกเม็นต์จะผ่านการทดสอบสเต็นซิลก็ต่อเมื่อ เมื่อเปรียบเทียบค่าอ้างอิงกับค่าที่อยู่

ในสเต็นซิลบัฟเฟอร์ ณ ตำแหน่งที่แฟรกเม็นต์นั้นอยู่แล้วเงื่อนไขตรงตามสเต็นซิลฟังก์ชัน เช่น กำหนดให้ฟังก์ชันของการทดสอบสเต็นซิลคือน้อยกว่าแฟรกเม็นต์จะผ่านก็ต่อเมื่อค่าค่าอ้างอิงนั้นน้อยกว่าค่าที่อยู่ในสเต็นซิลบัฟเฟอร์ แต่ก่อนที่จะมีการทดสอบใด ๆ ทั้งค่าอ้างอิงและค่าในสเต็นซิลบัฟเฟอร์จะต้องถูกแอนด์บิทด้วยมาสก์ก่อน

การดำเนินการสเต็นซิลมีหน้าที่ในการกำหนดการเปลี่ยนแปลงค่าในสเต็นซิลบัฟเฟอร์ในแต่ละกรณีตามที่ผู้ใช้กำหนด โดยมี 3 กรณีคือ กรณีที่ 1 แฟรกเม็นต์ไม่ผ่านการทดสอบสเต็นซิล กรณีที่ 2 แฟรกเม็นต์ผ่านการทดสอบสเต็นซิลแต่ไม่ผ่านการทดสอบความลึก กรณีที่ 3 แฟรกเม็นต์ผ่านการทดสอบสเต็นซิลและผ่านการทดสอบความลึก โดยวิธีการเปลี่ยนแปลงค่าที่ผู้ใช้สามารถกำหนดให้แต่ละกรณี คือ คงค่าเดิม เปลี่ยนเป็นศูนย์ แทนที่ด้วยค่าอ้างอิง เพิ่มหนึ่ง ลดหนึ่ง และผกผัน (invert)

2.2.3 **เท็กซ์เจอร์แม็พพิง** (Texture Mapping)

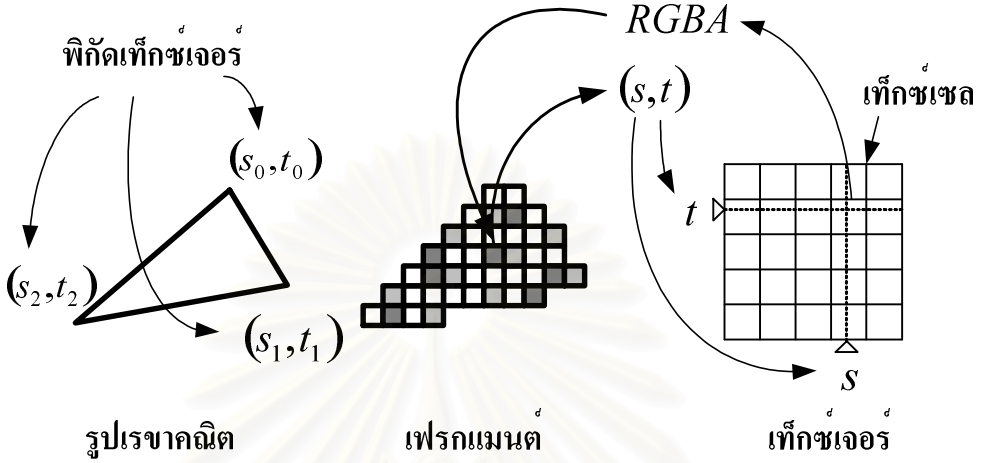
การทำเท็กซ์เจอร์แม็พพิงเป็นความสามารถในการนำข้อมูลจุดภาพในลักษณะที่อยู่ในรูปแบบของแถวลำดับมาวาดลงบนพื้นผิวเพื่อเพิ่มความสวยงามและความสมจริงให้กับพื้นผิวนั้น ดังตัวอย่างในรูปที่ 2.7 โดยตำแหน่งของจุดภาพที่นำมาวาดลงบนพื้นผิวจะถูกกำหนดโดยใช้ค่าพิกัดเท็กซ์เจอร์ที่อยู่ที่จุดยอดแต่ละจุดของพื้นผิว ซึ่งจะมีค่าระหว่าง 0 ถึง 1 ไม่ว่าจำนวนสมาชิกของแถวลำดับที่เป็นเท็กซ์เจอร์นั้นมีจำนวนเท่าไรก็ตาม พิกัดเท็กซ์เจอร์ที่ใช้ในการอ้างอิงตำแหน่งนั้นจะประกอบไปด้วยจำนวนของตัวเลขเท่ากับมิติของเท็กซ์เจอร์นั้นเท็กซ์เจอร์ ซึ่งมีอยู่ 3 แบบคือ เท็กซ์เจอร์ 1 มิติ เท็กซ์เจอร์ 2 มิติ และเท็กซ์เจอร์ 3 มิติ โดยแกนของเท็กซ์เจอร์จะแทนด้วยตัว s t และ r



รูปที่ 2.7 ตัวอย่างการทำเท็กซ์เจอร์แม็พพิง

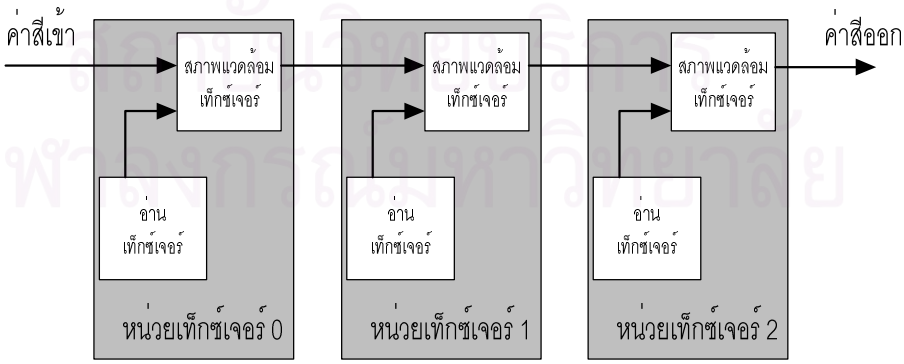
การทำเท็กซ์เจอร์แม็พพิงในขั้นตอนแรสเตอร์ไรเซชันของกราฟิกส์ฮาร์ดแวรมีขั้นตอนการทำงานดังรูปที่ 2.8 เริ่มจากพิกัดเท็กซ์เจอร์ที่จุดยอดของรูปเรขาคณิตพื้นฐานจะถูกประมาณค่าตามตำแหน่งของแต่ละแฟรกเม็นต์บนพื้นผิว จากนั้นแต่ละแฟรกเม็นต์จะนำค่าพิกัดเท็กซ์เจอร์ที่ได้ไปอ่านค่าสีจากเท็กซ์เจอร์ แต่เนื่องจากเท็กซ์เจอร์ก็เป็นเพียงคูลำดับของจุดชักตัวอย่างที่ไม่ต่อเนื่อง ดังนั้นจึงต้องใช้การประกอบให้คืนสภาพเพื่อประมาณค่าสีของเท็กซ์เจอร์ ณ ตำแหน่งพิกัดเท็กซ์เจอร์นั้นจากจุดชักตัวอย่าง

ของเท็กซ์เจอร์ซึ่งเรียกว่าเท็กซ์เซล (texel - texture element) ที่อยู่ข้างเคียง ซึ่งโดยทั่วไปจะใช้การประมาณค่าในช่วงแบบเชิงเส้น เมื่อได้ค่าสีจากการประมาณค่าแล้วจึงนำไปรวมกับค่าสีของแฟรกเม็นต์ซึ่งได้จากการประมาณค่าของค่าสีที่จุดยอดในขั้นตอนการทำแฟรกเม็นต์เชดดิงต่อไป



รูปที่ 2.8 ขั้นตอนการทำเท็กซ์เจอร์แม็พพิง

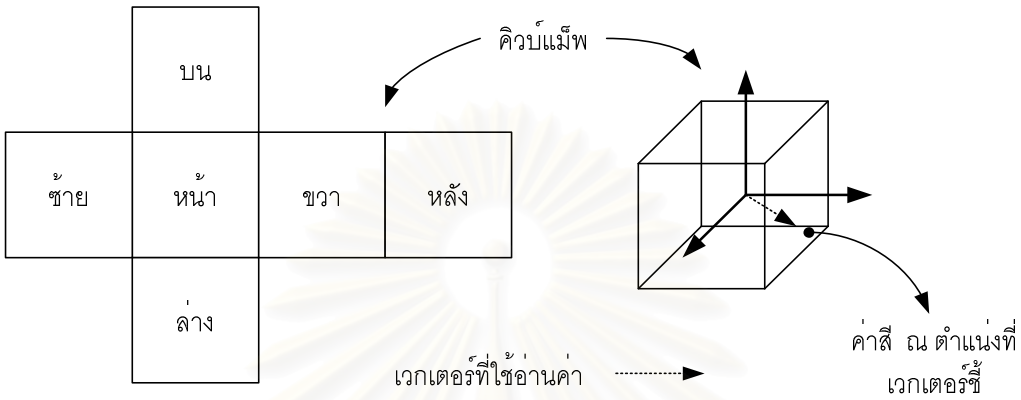
ในบางครั้งเพื่อเพิ่มความสมจริงของพื้นผิวให้มากขึ้น เท็กซ์เจอร์หลาย ๆ เท็กซ์เจอร์อาจจะถูกแม็พลงบนพื้นผิวเดียวกัน ซึ่งเราเรียกการทำเช่นนี้ว่า การทำมัลติเท็กซ์เจอริง (multi-texturing) ซึ่งมีขั้นตอนการทำงานดังรูปที่ 2.9 ค่าสีเข้าซึ่งคือค่าสีของแฟรกเม็นต์ซึ่งได้จากการประมาณค่าของค่าสีที่จุดยอดถูกส่งผ่านเข้าไปยังหน่วยเท็กซ์เจอร์ (จำนวนของหน่วยเท็กซ์เจอร์ที่มีจะขึ้นอยู่กับกราฟิกส์ฮาร์ดแวร์) ที่หน่วยเท็กซ์เจอร์แต่ละหน่วยจะทำการอ่านค่าเท็กซ์เจอร์แต่ละเท็กซ์เจอร์ขึ้นมา สภาพแวดล้อมเท็กซ์เจอร์ (texture environment) จะเป็นตัวกำหนดวิธีการรวมกันของสีเข้ามายังหน่วยเท็กซ์เจอร์กับค่าสีที่อ่านได้จากเท็กซ์เจอร์ภายในหน่วยเท็กซ์เจอร์นั้น ซึ่งจะมีฟังก์ชันที่ใช้ในการรวมสีไม่มากนัก เช่น การคูณ การทำอัลฟาเบล็นดิง เป็นต้น



รูปที่ 2.9 ขั้นตอนการทำมัลติเท็กซ์เจอริง

นอกจากนั้นยังมีเท็กซ์เจอร์อีกประเภทหนึ่งที่ถูกใช้ในงานวิจัยนี้ คือ เท็กซ์เจอร์คิวบ์แม็พ เท็กซ์เจอร์ชนิดนี้มีลักษณะเป็นเท็กซ์เจอร์ 2 มิติ 6 เท็กซ์เจอร์ประกอบกันเป็นกล่องทรงลูกบาศก์ ดังรูปที่

2.10 การอ่านค่าเท็กซ์เจอร์นั้นจะใช้พิกัดเท็กซ์เจอร์จำนวน 3 แกนคือ s t และ r โดยเริ่มจากจุดศูนย์กลางของลูกบาศก์แล้วใช้ค่าพิกัดทั้งสามเป็นขนาดของเวกเตอร์ตามแกน x y และ z ตามลำดับ ค่าสีที่อ่านได้คือค่าสี ณ ตำแหน่งบนผิวของลูกบาศก์ที่เวกเตอร์ชี้ไป



รูปที่ 2.10 เท็กซ์เจอร์คิวบ์แม็พ

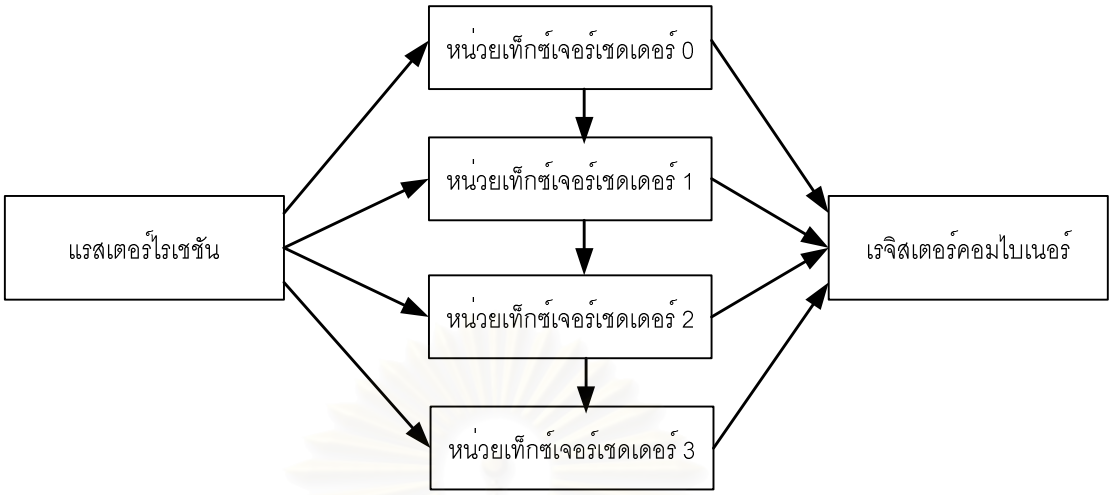
2.2.4 **แฟรกเม็นต์เชดดิงแบบโปรแกรมได้** (Programmable Fragment Shading)

จะเห็นได้ว่าขั้นตอนการทำเท็กซ์เจอร์แม็พพิงแบบมาตรฐานของโอเพ็นจีแอลในหัวข้อ 2.2.3 นั้นยังขาดความยืดหยุ่นอยู่มาก เช่น การอ่านเท็กซ์เจอร์ของแต่ละหน่วยไม่เกี่ยวข้องกัน ลำดับการทำงานตายตัว และจำนวนฟังก์ชันที่ใช้ได้ในสภาพแวดล้อมเท็กซ์เจอร์มีน้อย ผู้ผลิตกราฟิกส์ฮาร์ดแวร์แต่ละแห่งจึงพยายามที่จะเพิ่มความยืดหยุ่นของการทำงานในส่วนนี้ให้กับกราฟิกส์ฮาร์ดแวร์ของตนโดยโปรแกรมเมอร์สามารถใช้งานความสามารถที่เพิ่มเข้ามานี้ผ่านทาง ส่วนต่อขยายโอเพ็นจีแอล (OpenGL extension) มีผู้ผลิตหลักที่สร้างกราฟิกส์ฮาร์ดแวร์ที่มีความสามารถนี้ 2 บริษัทคือ บริษัทเอ็นวิเดีย (NVIDIA) และบริษัทเอทีไอ (ATI)

1) **เอ็นวิเดียแฟรกเม็นต์เชดดิง** (NVIDIA Fragment Shading) [7]

บริษัทเอ็นวิเดียได้เพื่อความยืดหยุ่นในการทำเท็กซ์เจอร์แม็พพิงโดยแบ่งการทำงานของเท็กซ์เจอร์แม็พพิงเป็น 2 ระยะ คือ ส่วนสำหรับการอ่านเท็กซ์เจอร์แบบโปรแกรมได้ซึ่งเรียกว่า เท็กซ์เจอร์แชดเดอร์ (texture shaders) และส่วนสำหรับการรวมสีแบบโปรแกรมได้เรียกว่า เรจิสเตอร์คอมไบเนอร์ (register combiners)

ในการใช้งานเท็กซ์เจอร์เชดเดอร์นั้นทำได้โดยผ่านส่วนต่อขยายโอเพ็นจีแอลจำนวน 3 ส่วน คือ GL_NV_texture_shader GL_NV_texture_shader2 และ GL_NV_texture_shader3 ซึ่งทั้ง 3 สามารถใช้งานได้บนกราฟิกส์ฮาร์ดแวร์ที่ใช้ชิปประมวลผลของเอ็นวิเดียรุ่น จีฟอร์ซ 4 (GeForce4) ขึ้นไป รูปที่ 2.11 ผังการทำงานของเท็กซ์เจอร์เชดเดอร์

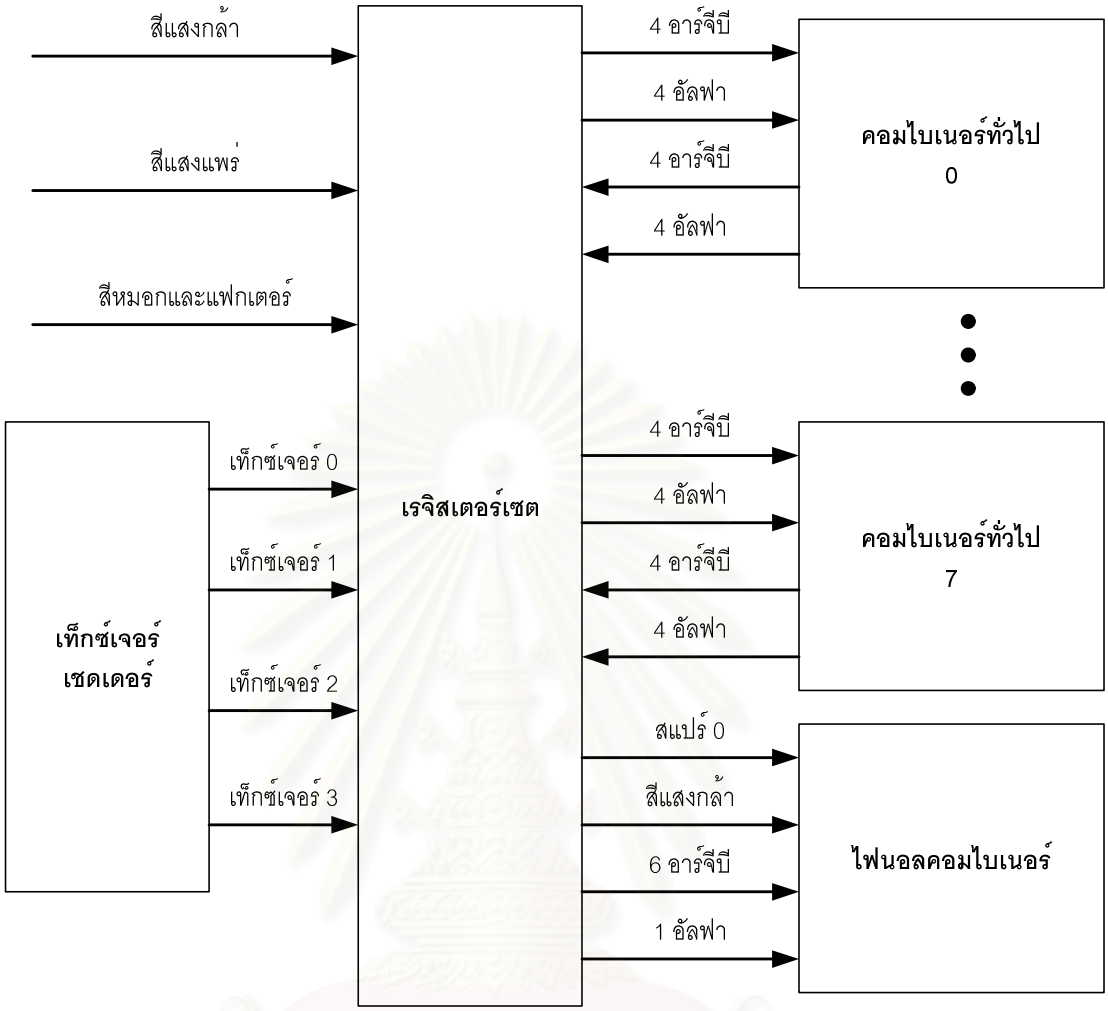


รูปที่ 2.11 การทำงานของเท็กซ์เจอร์เชดเดอร์

การทำงานของเท็กซ์เจอร์เชดเดอร์นั้นเริ่มจากพิกัดเท็กซ์เจอร์ของเท็กซ์เจอร์แต่ละตัวที่ถูกแม็พลงบนแฟรกเม็นต์ถูกส่งให้กับหน่วยเท็กซ์เจอร์เชดเดอร์แต่ละตัว (1 ตัวต่อ 1 เท็กซ์เจอร์) จากนั้นหน่วยเท็กซ์เจอร์ 0 จะเริ่มอ่านเท็กซ์เจอร์ตามโปรแกรมหนึ่งที่โปรแกรมเมอร์ทำการเลือกไว้จากทั้งหมด 37 โปรแกรม จากนั้นผลการอ่านค่าเท็กซ์เจอร์ของหน่วยที่ 0 จะถูกส่งให้กับทั้งหน่วยเท็กซ์เจอร์เชดเดอร์ถัดไป (หน่วยที่ 1) และส่วนการทำเรจิสเตอร์คอมไบเนอร์ จากนั้นหน่วยเท็กซ์เจอร์เชดเดอร์ที่ 1 จึงเริ่มอ่านเท็กซ์เจอร์ตามโปรแกรมที่โปรแกรมเมอร์ทำการเลือกไว้ซึ่งอาจจะเป็นคนละโปรแกรมกับของหน่วยเท็กซ์เจอร์เชดเดอร์ที่ 0 ก็ได้ ทำเช่นนี้จนกระทั่งครบถึงหน่วยเท็กซ์เจอร์เชดเดอร์ที่ 3 ซึ่งเป็นหน่วยเท็กซ์เจอร์เชดเดอร์สุดท้าย ข้อแตกต่างระหว่างการอ่านเท็กซ์เจอร์โดยใช้เท็กซ์เจอร์เชดเดอร์กับแบบมาตรฐานนอกเหนือจากการที่โปรแกรมเมอร์สามารถเลือกรูปแบบของการอ่านเท็กซ์เจอร์แล้วระหว่างผลลัพธ์ของหน่วยเท็กซ์เจอร์เชดเดอร์ก่อนหน้ายังสามารถส่งผลไปถึงการอ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์เชดเดอร์ถัดไปได้ ซึ่งการทำเช่นนี้เรียกว่า การอ่านเท็กซ์เจอร์แบบ ดีเพ็นแดนต์เท็กซ์เจอร์ริง (dependent texturing) ตัวอย่างการทำดีเพ็นแดนซ์เท็กซ์เจอร์ริงที่ใช้ในงานวิจัยนี้ได้แก่การใช้ค่าสีที่ได้จากการอ่านเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ก่อนหน้าเป็นพิกัดเท็กซ์เจอร์ในการอ่านเท็กซ์เจอร์ของหน่วยถัดไป

หลังจากจบการทำงานของเท็กซ์เจอร์เชดเดอร์แล้วค่าสีที่ได้จากการอ่านเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์แต่ละหน่วยจะถูกส่งต่อไปยังขั้นตอนการทำเรจิสเตอร์คอมไบเนอร์โดยจะถูกเขียนลงบนเรจิสเตอร์สีเท็กซ์เจอร์ 0 1 2 และ 3 ตามหมายเลขของหน่วยเท็กซ์เจอร์

เรจิสเตอร์คอมไบเนอร์เป็นส่วนที่ทำให้โปรแกรมเมอร์สามารถโปรแกรมการรวมค่าสีภายในแฟรกเม็นต์ของกราฟิกส์ฮาร์ดแวร์ได้ การใช้งานเรจิสเตอร์คอมไบเนอร์นั้นทำได้โดยผ่านส่วนต่อขยายโอเพ็นจีแอลจำนวน 2 ส่วน คือ GL_NV_register_combiners และ GL_NV_register_combiner2 รูปที่ 2.12 แสดงผังการทำงานของเรจิสเตอร์คอมไบเนอร์



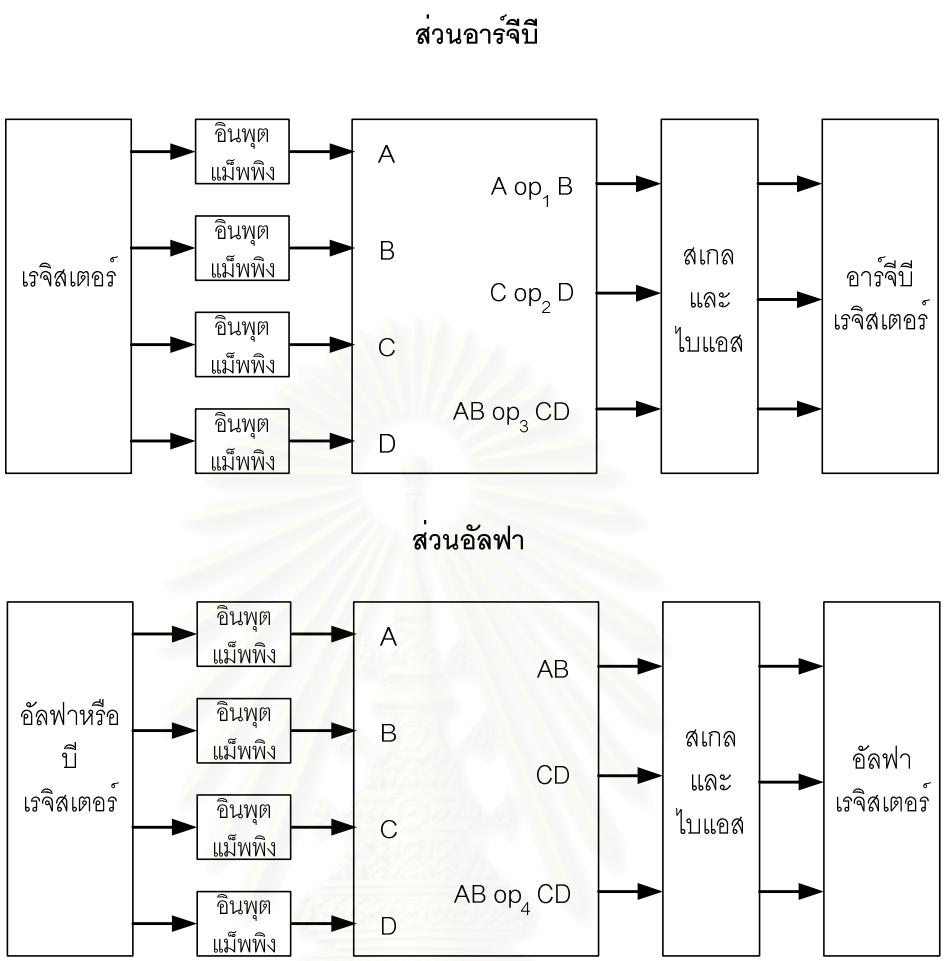
รูปที่ 2.12 การทำงานของเรจิสเตอร์คอมไบเนอร์

เรจิสเตอร์คอมไบเนอร์จะประกอบไปด้วยเรจิสเตอร์เซตซึ่งภายในจะมีเรจิสเตอร์จำนวน 13 ตัว ดังตารางที่ 2.1 โดยแต่ละตัวจะเก็บค่าอาร์จีบีและอัลฟา (ยกเว้นเรจิสเตอร์สีหมอกและแฟกเตอร์ซึ่งมีเฉพาะอาร์จีบี) และตัวคอมไบเนอร์ทั่วไปซึ่งมีหน้าที่ในการรวมสีจำนวน 8 ตัว (0-7) และไฟนอลคอมไบเนอร์อีก 1 ตัว การทำงานของเรจิสเตอร์คอมไบเนอร์จะเริ่มจากสีของแสงต่าง ๆ รวมถึงสีของหมอกจากขั้นตอนแรสเตอร์ไรเซชัน และสีที่ได้จากการอ่านเท็กซ์เจอร์ของเท็กซ์เจอร์เชดเดอร์ ถูกเขียนลงบนเรจิสเตอร์ จากนั้นคอมไบเนอร์ทั่วไปตัวที่ 0 จะเริ่มทำงานโดยทำการรวมค่าสีในรีจิสเตอร์เซตตามที่โปรแกรมเมอร์กำหนดผลลัพธ์ที่ได้จะถูกกลับมาเขียนลงบนเรจิสเตอร์เซตอีกครั้งหนึ่ง จากนั้นคอมไบเนอร์ตัวถัดไปก็ทำงาน เมื่อคอมไบเนอร์ทั่วไปตัวสุดท้ายคือตัวที่ 7 ทำงานเสร็จแล้ว ไฟนอลคอมไบเนอร์จึงทำงานและผลลัพธ์ของไฟนอลคอมไปเนอร์จะถูกส่งไปยังขั้นตอนการดำเนินการแฟรกเม็นต์ต่อไป

ตารางที่ 2.1 เรจิสเตอร์ภายในเรจิสเตอร์เซตของเรจิสเตอร์คอมไบเนอร์

เรจิสเตอร์	ชื่อ	อ่าน	เขียน
สีหลัก	col0	ได้	ได้
สีสีรอง	col1	ได้	ได้
สีเท็กซ์เจอร์ 0	tex0	ได้	ได้
สีเท็กซ์เจอร์ 1	tex1	ได้	ได้
สีเท็กซ์เจอร์ 2	tex2	ได้	ได้
สีเท็กซ์เจอร์ 3	tex3	ได้	ได้
สแปร์ 0	spare0	ได้	ได้
สแปร์ 1	spare1	ได้	ได้
สีคงตัว 0	const0	ได้	ไม่ได้
สีคงตัว 1	const1	ได้	ไม่ได้
สีหมอกและแฟกเตอร์	fog	ได้เฉพาะอาร์จีบี	ไม่ได้
ศูนย์	zero	ได้	ไม่ได้
ทิ้ง	discard	ไม่ได้	ได้

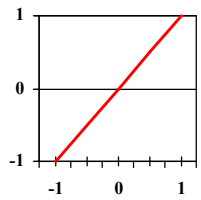
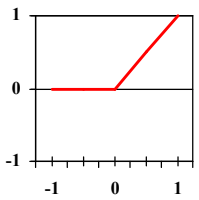
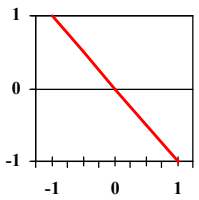
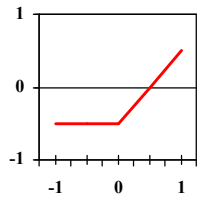
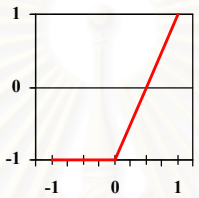
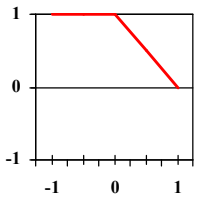
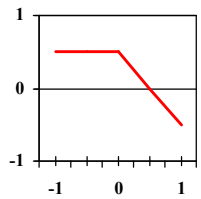
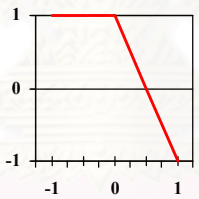
ภายในตัวคอมไบเนอร์ทั่วไปแต่ละตัวจะมีโครงสร้างดังรูปที่ 2.13 โดยแยกออกเป็น 2 ส่วนคือ ส่วนอาร์จีบีและส่วนอัลฟา โดยสามารถนำเข้าข้อมูลมาได้ครั้งละ 4 เรจิสเตอร์ตามที่โปรแกรมเมอร์กำหนดข้อมูลนำเข้าจะผ่านอินพุตแม็พพิงเพื่อปรับค่าให้เหมาะสม ซึ่งโปรแกรมเมอร์จะสามารถเลือกอินพุตแม็พพิงให้กับข้อมูลนำเข้าแต่ละตัวได้อย่างอิสระจากอินพุตแม็พพิงในรูปที่ 2.14 จากนั้นข้อมูลนำเข้าจะผ่านการดำเนินการของตัวดำเนินการ op_1 op_2 op_3 ซึ่งโปรแกรมเมอร์สามารถเลือกได้จากหนึ่งใน 5 รูปแบบในตารางที่ 2.2 โดยการรวมคือการรวมผลที่ได้จาก op_1 และ op_2 และตัวดำเนินการมักซ์ (mux) คือการทดสอบอัลฟาของเรจิสเตอร์ สแปร์ 0 ถ้ามีค่าน้อยกว่าหรือเท่ากับ 0.5 มักซ์จะเท่ากับผลลัพธ์ของ op_1 แต่ถ้ามากกว่ามักซ์จะเท่ากับผลลัพธ์ของ op_2 หลังจากนั้นผลลัพธ์ที่จากตัวดำเนินการทั้ง 3 จะผ่านปรับสเกลและไบแอสที่เลือกโดยโปรแกรมเมอร์จากหนึ่งในรูปที่ 2.15 ก่อนที่จะถูกเขียนลงบนเรจิสเตอร์ต่อไป



รูปที่ 2.13 โครงสร้างของตัวคอมไบเนอร์ทั่วไป

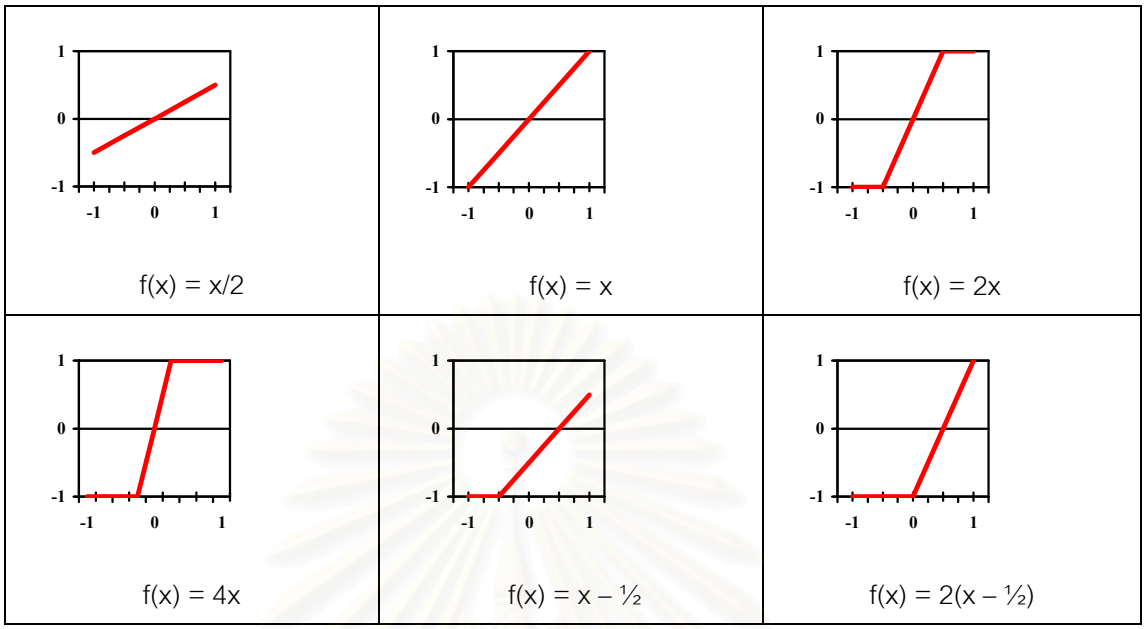
ตารางที่ 2.2 รูปแบบของตัวดำเนินการที่ใช้ได้ในตัวคอมไบเนอร์ทั่วไป

รูปแบบ	op1	op2	op3
1	คูณจุด	คูณจุด	ทิ้ง
2	คูณจุด	คูณ	ทิ้ง
3	คูณ	คูณจุด	ทิ้ง
4	คูณ	คูณ	รวม
5	คูณ	คูณ	มักซ์

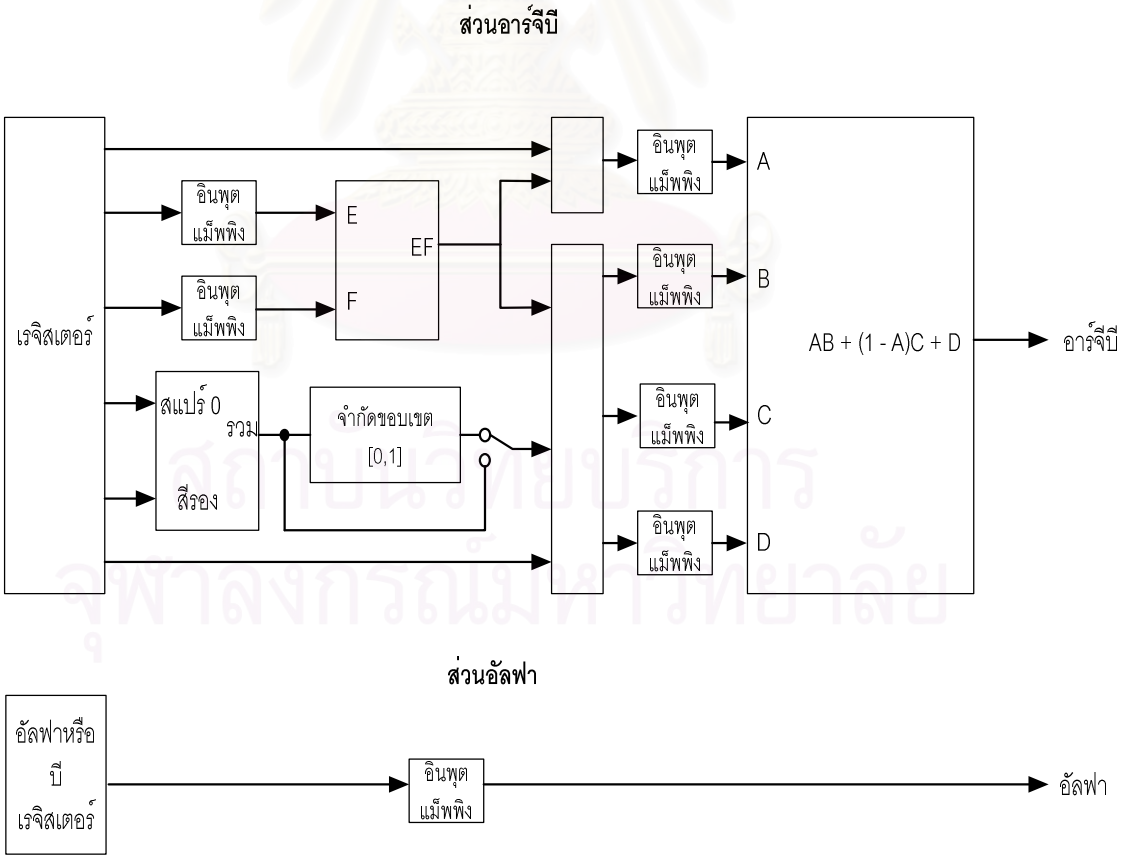
Signed Identity  f(x) = x	Unsigned Identity  f(x) = max(0, x)	Signed Negate  f(x) = -x
Half Bias Normal  f(x) = max(0, x) – ½	Expand Normal  f(x) = 2 * max(0, x) - 1	Unsigned Invert  f(x) = 1-min(max(0,x),1)
Half Bias Negate  f(x) = -max(0, x) + ½	Expand Negate  f(x) = -2 * max(0, x) + 1	

รูปที่ 2.14 อินพุตแม็พพิงของตัวคอมไบเนอร์ทั่วไป

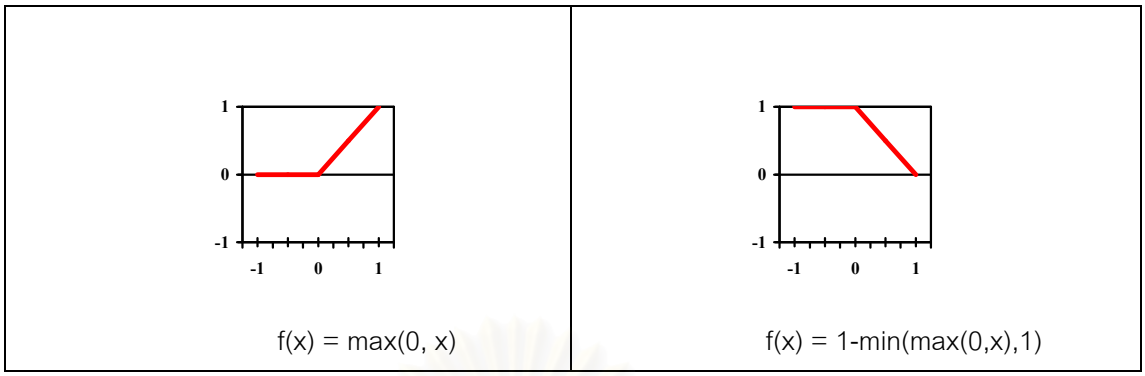
ตัวไฟนอลคอมไบเนอร์จะแตกต่างจากตัวคอมไบเนอร์ทั่วไปโดยฟังก์ชันการทำงานของไฟนอลคอมไบเนอร์จะเป็นแบบตายตัวไม่สามารถเปลี่ยนแปลงได้ รูปที่ 2.16 แสดงโครงสร้างภายในของไฟนอลคอมไบเนอร์ โปรแกรมเมอร์จะสามารถเลือกได้เพียงเรจิสเตอร์นำเข้าและอินพุตแม็พพิงซึ่งอินพุตแม็พพิงนี้สามารถเลือกได้เพียง 2 รูปแบบในรูป 2.16 เท่านั้น และผลลัพธ์ของไฟนอลคอมไบเนอร์จะไม่มีการสเกลและไบแอส แต่จะถูกส่งไปยังขั้นตอนการดำเนินการแฟรกเม็นต์โดยตรง



รูปที่ 2.15 สเกลและไบแอสของตัวคอมไบเนอร์ทั่วไป



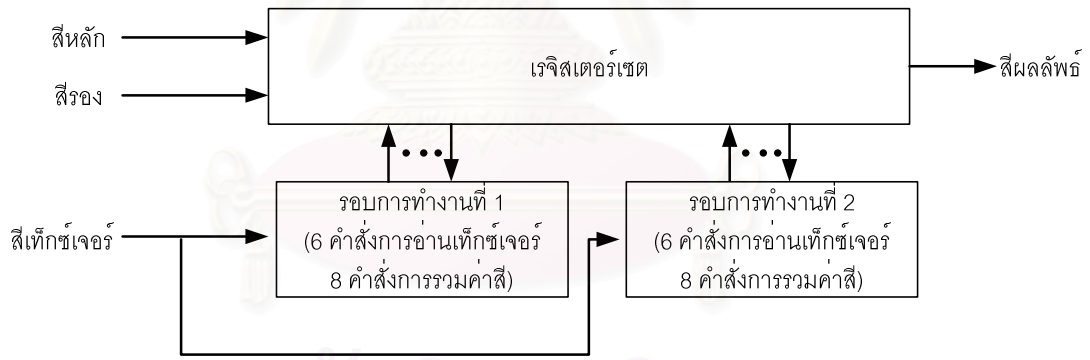
รูปที่ 2.16 โครงสร้างของตัวไฟนอลคอมไบเนอร์



รูปที่ 2.17 อินพุตแม็พพิงของไฟนอลคอมไบเนอร์

2) เอทีไอแฟรกเม็นต์เชดดิง (ATI Fragment Shading) [8]

แฟรกเม็นต์เชดดิงของเอทีไอจะไม่แบ่งการทำงานออกเป็น 2 สถานะเหมือนกับของเอ็นวิเดียแต่รวมทั้งการอ่านเท็กซ์เจอร์และการรวมสีเข้าไว้ด้วยกัน โดยการใช้งานสามารถทำได้โดยผ่านส่วนต่อขยายโอเพ็นจีแอลเพียงส่วนเดียว คือ GL_ATI_fragment_shader ซึ่งสนับสนุนโดยกราฟิกส์ฮาร์ดแวร์ของบริษัทเอทีไอรุ่นราเดออน 8500 (Radeon 8500) ขึ้นไป รูปที่ 2.17 แสดงผังการทำงานของเอทีไอแฟรกเม็นต์เชดดิง



รูปที่ 2.18 การทำงานของเอทีไอแฟรกเม็นต์เชดดิง

บนกราฟิกส์ฮาร์ดแวร์เอทีไอราเดออน 8500 แฟรกเม็นต์เชดดิงจะแบ่งการทำงานออกเป็น 2 รอบการทำงานโดยภายในแต่ละรอบการทำงานจะเริ่มด้วยคำสั่งที่ใช้อ่านค่าสีจากเท็กซ์เจอร์ แล้วตามด้วยคำสั่งสำหรับการรวมค่าสี ซึ่งคำสั่งเหล่านี้จะทำการอ่านและแก้ไขค่าสีในเรจิสเตอร์เซตซึ่งประกอบไปด้วยเรจิสเตอร์ดังตารางที่ 2.3 จำนวนคำสั่งที่ใช้ในการอ่านค่าสีจากเท็กซ์เจอร์ในแต่ละรอบจะต้องไม่เกินจำนวนของหน่วยเท็กซ์เจอร์ที่มีในกราฟิกส์ฮาร์ดแวร์ (6 สำหรับราเดออน 8500) และจำนวนคำสั่งในการรวมค่าสีต้องไม่เกิน 8 การทำงานของแฟรกเม็นต์เชดดิงจะเริ่มจากค่าสีหลัก (primary color) และค่าสีรอง (secondary color) ที่ได้จากขั้นตอนการประมาณค่าสีในขบวนการแรสเตอร์ไรเซชันถูกเขียนลงบน

เรจิสเตอร์ จากนั้นกราฟิกส์ฮาร์ดแวร์จะดำเนินการตามคำสั่งที่อยู่ภายในรอบการทำงานที่ 1 และ 2 ตามลำดับ ค่าสีผลลัพธ์ที่จะถูกส่งต่อไปยังขั้นตอนการดำเนินการแฟรกเม็นต์คือค่าสีสุดท้ายของเรจิสเตอร์ 0

ตารางที่ 2.3 เรจิสเตอร์เซตของเอทีไอแฟรกเม็นต์เชดดิง

เรจิสเตอร์	**ชื่อ**	**อ่าน**	**เขียน**
สีหลัก	primary color	ได้	ไม่ได้
สีรอง	secondary color	ได้เฉพาะอาร์จีบี	ไม่ได้
เรจิสเตอร์ 0	reg0	ได้	ได้
เรจิสเตอร์ 1	reg1	ได้	ได้
เรจิสเตอร์ 2	reg2	ได้	ได้
เรจิสเตอร์ 3	reg3	ได้	ได้
เรจิสเตอร์ 4	reg4	ได้	ได้
เรจิสเตอร์ 5	reg5	ได้	ได้
สีคงตัว 0	con0	ได้	ไม่ได้
สีคงตัว 1	con1	ได้	ไม่ได้
สีคงตัว 2	con2	ได้	ไม่ได้
สีคงตัว 3	con3	ได้	ไม่ได้
สีคงตัว 4	con4	ได้	ไม่ได้
สีคงตัว 5	con5	ได้	ไม่ได้
สีคงตัว 6	con6	ได้	ไม่ได้
สีคงตัว 7	con7	ได้	ไม่ได้

ชุดคำสั่งที่ใช้ในการรวมสีของเอทีไอแฟรกเม็นต์เชดชิง ได้แก่

- MOV ย้ายค่าสีระหว่างเรจิสเตอร์
- ADD รวมค่าสีระหว่างเรจิสเตอร์ 2 เรจิสเตอร์แล้วเก็บไว้ในเรจิสเตอร์ที่ 3
- SUB ลบค่าสีระหว่างเรจิสเตอร์ 2 เรจิสเตอร์แล้วเก็บไว้ในเรจิสเตอร์ที่ 3
- MUL คูณค่าสีระหว่างเรจิสเตอร์ 2 เรจิสเตอร์แล้วเก็บไว้ในเรจิสเตอร์ที่ 3
- MAD คูณค่าสีระหว่างเรจิสเตอร์ 2 เรจิสเตอร์ จากนั้นบวกกับค่าสีในเรจิสเตอร์ที่ 3 แล้วเก็บในเรจิสเตอร์ที่ 4
- LERP ทำการประมาณค่าแบบเชิงเส้นระหว่างเรจิสเตอร์ 2 เรจิสเตอร์ โดยให้ค่าใน

เรจิสเตอร์ที่ 3 เป็นน้ำหนักที่ใช้ในการประมาณค่า ผลที่ได้เก็บในเรจิสเตอร์ที่ 4

- DOT3 ทำการคูณจุดแบบ 3 ส่วนประกอบ ระหว่างเรจิสเตอร์ 2 เรจิสเตอร์แล้วเก็บไว้ในเรจิสเตอร์ที่ 3
- DOT4 ทำการคูณจุดแบบ 4 ส่วนประกอบ ระหว่างเรจิสเตอร์ 2 เรจิสเตอร์แล้วเก็บไว้ในเรจิสเตอร์ที่ 3
- DOT2_ADD เหมือน DOT3 เพียงแต่กำหนดตัวประกอบตัวที่ 3 เท่ากับ 1.0 ดังนั้นจึงเสมือนว่าไม่ได้ทำคูณ
- CND ย้ายค่าจากเรจิสเตอร์ตัวที่ 1 หรือ 2 ไปใส่ไว้ในเรจิสเตอร์ 3 ขึ้นอยู่กับค่าในเรจิสเตอร์ตัวที่ 4 ว่ามากกว่า 0.5 หรือไม่
- CND0 เหมือนกับ CND เพียงแต่เปรียบเทียบกับค่า 0.0

ตัวอย่างการโปรแกรมแฟรกเม็นต์เชดดิงแสดงในรูปที่ 2.19 บรรทัดที่ 3 และ 4 เป็นคำสั่งในการอ่านค่าเท็กซ์เจอร์ในรอบแรก บรรทัดที่ 6-9 เป็นคำสั่งในการย้ายค่าจากเรจิสเตอร์ 5 มาเก็บไว้ในเรจิสเตอร์ 2 บรรทัดที่ 15 เป็นการอ่านค่าเท็กซ์เจอร์ในรอบที่ 2

```
1   glBeginFragmentShaderATI();
2
3     glSampleMapATI(GL_REG_0_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Pri. Volume Data
4     glSampleMapATI(GL_REG_5_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Sec.
5
6     glColorFragmentOp1ATI(GL_MOV_ATI,
7               GL_REG_2_ATI, GL_NONE, GL_NONE,
8               GL_REG_5_ATI, GL_RED, GL_NONE);
9
10    glColorFragmentOp1ATI(GL_MOV_ATI,
11              GL_REG_2_ATI, GL_RED_BIT_ATI, GL_NONE,
12              GL_REG_0_ATI, GL_ALPHA, GL_NONE);
13
14    //End of first pass
15    glSampleMapATI(GL_REG_1_ATI, GL_REG_2_ATI,GL_SWIZZLE_STR_ATI); //Sample TF
16
17
18    glColorFragmentOp1ATI(GL_MOV_ATI,
19              GL_REG_0_ATI, GL_NONE, GL_NONE,
20              GL_REG_1_ATI, GL_NONE, GL_NONE);
21
22    glAlphaFragmentOp1ATI(GL_MOV_ATI,
23              GL_REG_0_ATI, GL_NONE,
24              GL_REG_1_ATI, GL_ALPHA, GL_NONE);
25
26
27  glEndFragmentShaderATI();
```

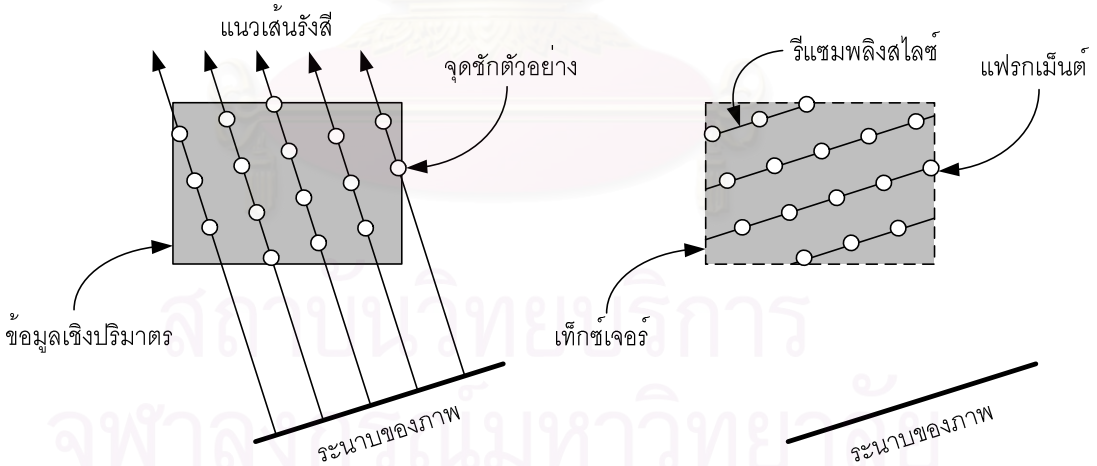
รูปที่ 2.19 ตัวอย่างการโปรแกรมแฟรกเม็นต์เชดดิง

2.3 การสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์สามมิติ (Volume Rendering via 3D Texture Mapping) [9,10]

ความคล้ายกันของขั้นตอนการสร้างภาพเชิงปริมาตรและขั้นตอนการแม็พเท็กซ์เจอร์คือทั้งคู่ต้องมีการประกอบให้คืนสภาพจากจุดชักตัวอย่างเดิมแล้วชักตัวอย่างที่ตำแหน่งใหม่ โดยจุดชักตัวอย่างเดิม

ของทั้งคู่หรือจุดปริมาตรและเท็กซ์เซลนั้นต่างก็ถูกเก็บอยู่ในรูปแบบของแถวลำดับเหมือนกัน โดยเฉพาะในกรณีที่เป็นการแม็พเท็กซ์เจอร์สามมิติซึ่งเท็กซ์เซลถูกเก็บในแถวลำดับสามมิติเหมือนกับจุดปริมาตรของข้อมูลเชิงปริมาตร การแม็พเท็กซ์เจอร์จึงสามารถใช้แทนการชักตัวอย่างในการสร้างภาพเชิงปริมาตรได้ รวมถึงหลังจากขั้นตอนการชักตัวอย่างแล้วกราฟิกส์ฮาร์ดแวร์ยังสนับสนุนการรวมค่าสีด้วยอัลฟาเบล็นดิงซึ่งเป็นวิธีเดียวกันกับที่ใช้ในการรวมค่าสีของจุดชักตัวอย่างในการสร้างภาพเชิงปริมาตร จึงมีการนำการกราฟิกส์ฮาร์ดแวร์มาใช้ในการสร้างภาพเชิงปริมาตร เมื่อขั้นตอนที่ใช้เวลาในการประมวลผลสูงทั้งการประกอบให้คืนสภาพ การชักตัวอย่างและการรวมค่าสีทำโดยกราฟิกส์ฮาร์ดแวร์ ซึ่งปัจจุบันมีประสิทธิภาพการทำงานที่สูงจึงทำให้การสร้างภาพเชิงปริมาตรโดยวิธีนี้มีประสิทธิภาพที่สูงกว่าวิธีใช้ซอฟต์แวร์เป็นอย่างมาก

ขั้นตอนของการสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์สามมิติเริ่มจากแปลงข้อมูลเชิงปริมาตรให้เป็นเท็กซ์เจอร์สามมิติโดยแปลงจุดปริมาตรแต่ละจุดให้เป็นเท็กซ์เซล จากนั้นทำการชักตัวอย่างจากเท็กซ์เจอร์นั้นโดยการนำไปแม็พลงบนพื้นผิวซึ่งสร้างให้ขนานกับระนาบของภาพผลลัพธ์ที่ต้องการสร้างขึ้นและมีขอบเขตอยู่ภายเท็กซ์เจอร์นั้นเรียกพื้นผิวนี้ว่ารีแซมพลิงสไลซ์ รูปที่ 2.20 เปรียบเทียบระหว่างการชักตัวอย่างแบบวิธีการฉายแสงและการชักตัวอย่างโดยใช้การแม็พเท็กซ์เจอร์ โดยสมมติให้ตำแหน่งของผู้สังเกตอยู่ไกลจากระนาบของภาพผลลัพธ์มากจนประมาณได้ว่าแนวรังสีที่ใช้ชักตัวอย่างเป็นรังสีขนาน



(ก) วิธีการฉายแสง *(ข) วิธีการแม็พเท็กซ์เจอร์*

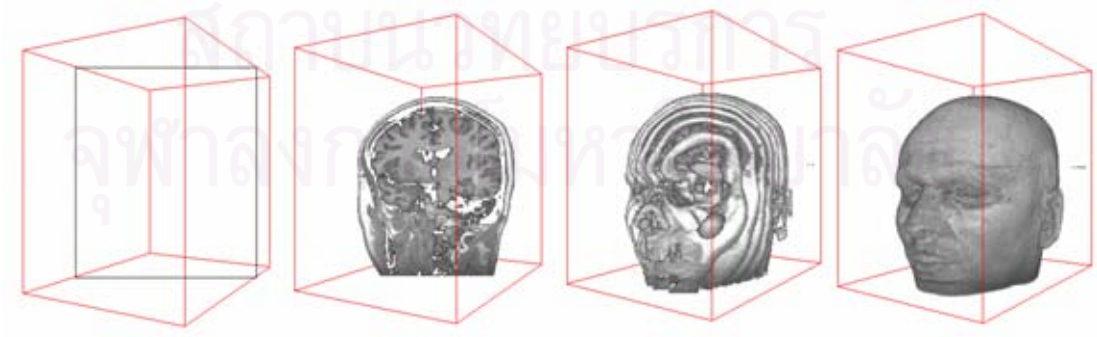
รูปที่ 2.20 การชักตัวอย่างในการสร้างภาพเชิงปริมาตร

ในขั้นตอนแรสเตอร์ไรเซชันรีแซมพลิงสไลซ์จะถูกแตกออกเป็นแฟรกเม็นต์จากนั้นแฟรกเม็นต์แต่ละตัวจะเข้าสู่ขบวนการทำเท็กซ์เจอร์แม็พพิงโดยกราฟิกส์ฮาร์ดแวร์จะทำการชักตัวอย่างค่าสีจากการประกอบให้คืนภาพของเท็กซ์เจอร์ที่ตำแหน่งพิกัดเท็กซ์เจอร์ของแฟรกเม็นต์นั้นค่าสีที่ได้จะถูกนำไปรวม

กับค่าสีเดิมของแฟรกเม็นต์ซึ่งโดยทั่วไปแล้วจะกำหนดให้เป็นศูนย์ หรือไม่ก็กำหนดให้สภาพแวดล้อมเท็กซ์เจอร์ทำการรวมแบบแทนที่ (replace) คือใช้ค่าสีของเท็กซ์เจอร์แทนค่าสีเดิมโดยไม่ต้องทำการรวมค่าสี

หลังจากขั้นตอนการชักตัวอย่างแล้วขั้นตอนถัดไปในการสร้างภาพเชิงปริมาตรโดยวิธีการฉายแสงคือการรวมค่าสีที่ได้จากชักตัวอย่างที่อยู่บนแนวเส้นรังสีเดียวกันเข้าด้วยกันโดยใช้การทำอัลฟาเบล็นดิงตามสมการที่ 2.9 ในกรณีที่ทำการรวมจากหลังไปหน้า และสมการที่ 2.10 กับ 2.11 ในกรณีที่ทำการรวมจากหน้าไปหลัง ถึงแม้ว่าการชักตัวอย่างโดยการใช้รีแซมพลิงสไลซ์ของการแม็พเท็กซ์เจอร์ไม่ได้ทำตามแนวเส้นรังสีแต่การรวมค่าสีของแฟรกเม็นต์ของแต่ละรีแซมพลิงสไลซ์ที่อยู่บนแนวเส้นรังสีเดียวกันยังสามารถทำได้บนเฟรมบัฟเฟอร์ เนื่องจากรีแซมพลิงสไลซ์นั้นขนานกับระนาบของภาพผลลัพธ์ซึ่งก็คือระนาบของเฟรมบัฟเฟอร์ตำแหน่งบนแฟรมบัฟเฟอร์ของแฟรกเม็นต์ที่อยู่บนแนวเส้นรังสีเดียวกันจึงเป็นตำแหน่งเดียวกัน เมื่อเป็นเช่นนี้จึงสามารถใช้ความสามารถของกราฟิกส์ฮาร์ดแวร์ในการรวมค่าสีของแฟรกเม็นต์กับค่าสีเดิมบนเฟรมบัฟเฟอร์ที่ตำแหน่งเดียวกันโดยใช้อัลฟาเบล็นดิงในการรวมค่าสีของแฟรกเม็นต์ที่อยู่บนแนวเส้นรังสีเดียวกันนี้ได้ แต่เนื่องจากเฟรมบัฟเฟอร์ของกราฟิกส์ฮาร์ดแวร์ส่วนใหญ่ไม่สนับสนุนการทั้งเก็บและรวมค่าอัลฟาดังนั้นจึงไม่สามารถใช้การรวมค่าสีจากหน้าไปหลังได้ ส่งผลให้การวาดรีแซมพลิงสไลซ์ต้องทำจากหลังไปหน้าตามลำดับเท่านั้นและต้องกำหนดให้กราฟิกส์ฮาร์ดแวร์ทำการรวมค่าสีของแฟรกเม็นต์กับค่าสีบนเฟรมบัฟเฟอร์ตามสมการที่ 2.9

รูปที่ 2.21 เป็นการสรุปขั้นตอนการทำงานของการสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์สามมิติรูปที่ 2.21 ก เป็นรีแซมพลิงสไลซ์ภายในขอบเขตของเท็กซ์เจอร์ รูปที่ 2.21 ข เมื่อกราฟิกส์ฮาร์ดแวร์ทำการวาดเท็กซ์เจอร์ลงบนรีแซมพลิงสไลซ์ รูปที่ 2.21 ค เมื่อเพิ่มจำนวนรีแซมพลิงสไลซ์ให้มากขึ้นสังเกตการซ้อนทับบนตำแหน่งเดียวกันของแฟรกเม็นต์ที่อยู่ในแนวเดียวกัน รูปที่ 2.21 ง เพิ่มจำนวน รีแซมพลิงสไลซ์และกำหนดให้กราฟิกส์ฮาร์ดแวร์ทำการรวมค่าสีของแฟรกเม็นต์ที่ซ้อนทับกับโดยใช้สมการที่ 2.9



(ก) *(ข)* *(ค)* *(ง)*

รูปที่ 2.21 การสร้างภาพเชิงปริมาตรโดยใช้การทำเท็กซ์เจอร์แม็ปปิง

2.4 การให้แสงเงา (Shading) [11]

การให้แสงเงาเป็นเทคนิคในคอมพิวเตอร์กราฟิกส์ที่ช่วยให้ภาพที่ได้มีมิติสมจริงขึ้น แบบจำลองการสะท้อนแสงของ Phong (Phong reflection model) [4] เป็นแบบจำลองที่ถูกใช้ในการให้แสงเงาในงานคอมพิวเตอร์กราฟิกส์อย่างกว้างขวาง เนื่องจากให้คุณภาพของภาพที่ดี และใช้การคำนวณที่ไม่ยุ่งยาก โดยแบบจำลองนี้แบ่งการสะท้อนแสงออกเป็น 3 ชนิดคือ การสะท้อนแสงล้อมรอบ (ambient reflection) การสะท้อนแสงแพร่ (diffuse reflection) และการสะท้อนแสงกล้า (specular reflection) โดยที่จะสามารถรวมเข้าด้วยกันด้วยสมการที่ 2.13 และรูปที่ 2.22 จะแสดงผลของการสะท้อนแสงชนิดต่าง ๆ ต่อแสงเงาของวัตถุ

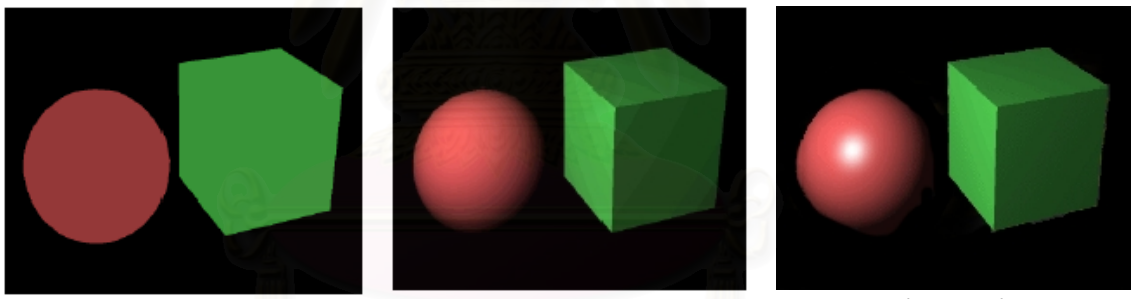
$$I = I_a + I_d + I_s \qquad \text{...(2.13)}$$

เมื่อ I คือค่าความเข้มแสงรวมที่วัตถุสะท้อนออกมา

I_a คือค่าความเข้มของแสงล้อมรอบที่วัตถุสะท้อนออกมา

I_d คือค่าความเข้มของแสงสะท้อนแบบกระจายที่วัตถุสะท้อนออกมา

I_s คือค่าความเข้มของแสงการสะท้อนแบบกระจกที่วัตถุสะท้อนออกมา



(ก) การสะท้อนแสงล้อมรอบ | *(ข) การสะท้อนแสงล้อมรอบรวมกับการสะท้อนแสงแพร่* | *(ค) การสะท้อนแสงล้อมรอบรวมกับการสะท้อนแสงแพร่และการสะท้อนแสงกล้า*

รูปที่ 2.22 ผลของการสะท้อนแสงชนิดต่าง ๆ ต่อภาพของวัตถุ

2.4.1 การสะท้อนแสงล้อมรอบ

แสงล้อมรอบคือแสงที่มาจากทุก ๆ ทิศทางเท่า ๆ กัน ไม่สามารถหาทิศทางของแสงได้ และเมื่อตกกระทบบนวัตถุแสงจึงสะท้อนออกไปทุกทิศทางด้วยเช่น แสงที่เกิดจากการสะท้อนของกำแพงหลาย ๆ ด้านในห้อง ไม่ว่า ณ ตำแหน่งที่พิจารณาให้แสงเงานั้นวัตถุจะมีเวกเตอร์ปกติของพื้นผิวไปทางทิศทางไหน หรือผู้สังเกตมองจากทิศทางไหน ความเข้มของแสงล้อมรอบที่วัตถุสะท้อนออกมาก็จะเท่ากัน โดยสามารถอธิบายด้วยสมการ 2

$$I_a = k_a I_A \qquad \text{...(2.14)}$$

เมื่อ I_a คือความเข้มของแสงล้อมรอบที่วัตถุสะท้อนออกมา

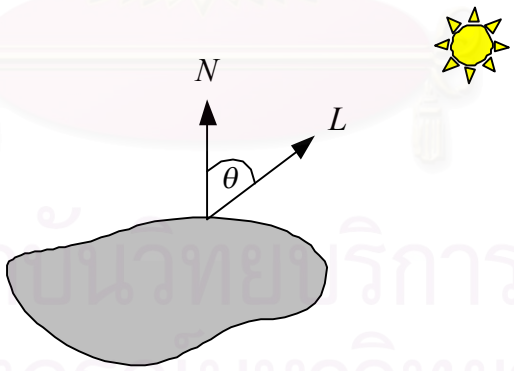
k_a คือสัมประสิทธิ์การสะท้อนแสงล้อมรอบของวัตถุ

I_A คือความเข้มของแสงล้อมรอบที่ส่องไปกระทบวัตถุ

2.4.2 การสะท้อนแสงแพร่

การสะท้อนแสงแพร่เกิดเมื่อแสงจากแหล่งกำเนิดแสงที่มีทิศทางแน่นอนส่องกระทบวัตถุแล้ววัตถุสะท้อนแสงกระจายออกไปทุกทิศทางเท่า ๆ กัน การสะท้อนแสงแบบนี้จะเกิดกับวัตถุที่มีผิวด้านโดยทั่วไปแล้วจะกำหนดให้แหล่งกำเนิดแสงเป็นจุดที่อยู่ไกลมาก ๆ เพื่อให้แสงที่ส่องมามีลักษณะเป็นรังสีขนานและความเข้มของแสงไม่ขึ้นกับระยะห่างของแหล่งกำเนิดแสงกับวัตถุ เนื่องจากอยู่ไกลมาก ๆ เพื่อให้ง่ายต่อการคำนวณ

ความเข้มของการสะท้อนแสงแบบนี้ขึ้นอยู่กับมุมที่เวกเตอร์ปกติของพื้นผิววัตถุ ณ ตำแหน่งนั้นหันเหออกไปจากทิศทางของแหล่งกำเนิดแสงมากน้อยเพียงไร ถ้าเวกเตอร์ปกติของพื้นผิวหันเหไปจากทิศของแหล่งกำเนิดแสงมากแสงที่สะท้อนออกมาจะมีความเข้มต่ำ และในทางกลับกันถ้าเวกเตอร์ปกติของพื้นผิวหันไปในทิศทางเดียวกับแหล่งกำเนิดแสงความเข้มของแสงที่สะท้อนออกมาจะมาก รูปที่ 2.23 แสดงองค์ประกอบต่าง ๆ ที่ใช้ในการคำนวณความเข้มของการสะท้อนแสงแพร่ของวัตถุ ซึ่งสามารถอธิบายด้วยสมการที่ 2.15



รูปที่ 2.23 องค์ประกอบการสะท้อนแสงแพร่

$$I_d = k_d I_i \cos\theta = k_d I_i (N \cdot L) \qquad ;0 \le \theta \le 2\pi \qquad \text{...(2.15)}$$

เมื่อ I_d คือความเข้มของการสะท้อนแสงแพร่ที่วัตถุสะท้อนออกมา

k_d คือสัมประสิทธิ์การสะท้อนแสงแพร่ของวัตถุ

I_i คือความเข้มแสงของแหล่งกำเนิด

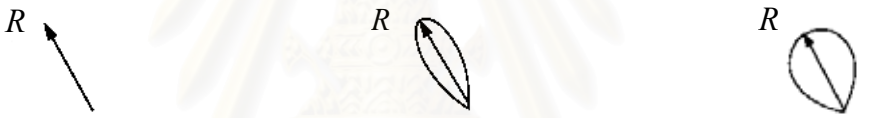
N คือเวกเตอร์ปกติของผิววัตถุ ณ ตำแหน่งนั้น

L คือเวกเตอร์ที่ชี้ไปยังแหล่งกำเนิดแสง

θ คือมุมที่เวกเตอร์ N กระทำกับเวกเตอร์ L

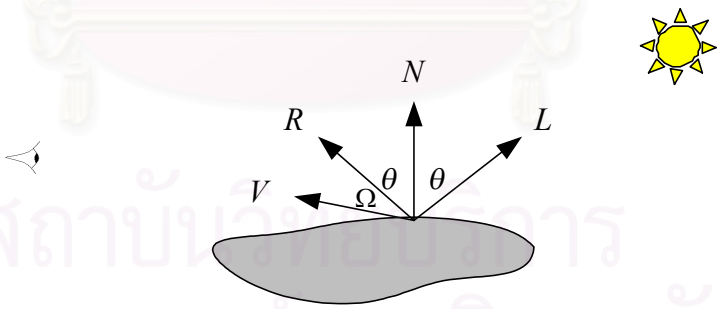
2.4.3 การสะท้อนแสงกล้า

การสะท้อนแสงกล้าเกิดเมื่อแสงจากแหล่งกำเนิดแสงที่มีทิศทางแน่นอนส่องกระทบวัตถุแล้ววัตถุสะท้อนแสงกระจายออกไปในทิศทางใดทิศทางหนึ่งที่แน่นอนโดยทิศทางจะขึ้นอยู่กับเวกเตอร์ปกติของพื้นผิวและเวกเตอร์ที่ชี้ไปยังแหล่งกำเนิดแสง แสงที่สะท้อนออกมาจะกระจายออกเป็นมุมกว้างหรือแคบขึ้นอยู่กับค่าดัชนีที่ใช้จำลองความเรียบมันของวัตถุ ดังรูปที่ 2.24 ในกรณีที่วัตถุเป็นตัวสะท้อนแสงอย่างสมบูรณ์ เช่น กระจกค่าดัชนีนี้จะเป็นอนันต์แสงที่สะท้อนจะออกไปในทิศทางเดียวเท่านั้น ในทางกลับกันถ้าวัตถุมีความเรียบมันน้อย (ค่าดัชนีน้อย) แสงก็จะสะท้อนออกไปในมุมกว้าง การสะท้อนแสงกล้านี้เกิดขึ้นกับวัตถุที่มีผิวเรียบมัน โดยค่าความเข้มของแสงที่สะท้อนออกมาขึ้นอยู่กับทิศทางของแสงที่สะท้อนออกมาจากวัตถุ และทิศทางที่ผู้สังเกตมองวัตถุ รูปที่ 2.25 จะแสดงองค์ประกอบต่าง ๆ ที่ใช้ในการคำนวณความเข้มของการสะท้อนแสงกล้าของวัตถุ สามารถอธิบายด้วยสมการที่ 2.16



(ก) $n = \infty$ (ข) $n = 100$ (ค) $n = 8$

รูปที่ 2.24 ค่าดัชนีที่ใช้จำลองความเรียบมันของวัตถุและมุมกระจายของแสงกล้า



รูปที่ 2.25 องค์ประกอบการสะท้อนแสงกล้า

$$I_s = k_s I_i \cos^n \Omega = k_s I_i (R \cdot V)^n \qquad \text{...(2.16)}$$

เมื่อ I_s คือความเข้มของการสะท้อนแสงกล้าที่วัตถุสะท้อนออกมา

k_s คือสัมประสิทธิ์การสะท้อนแสงแพร่ของวัตถุ

I_i คือความเข้มแสงของแหล่งกำเนิดแสง

N คือเวกเตอร์ปกติของผิววัตถุ ณ ตำแหน่งนั้น

L คือเวกเตอร์ที่ชี้ไปยังแหล่งกำเนิดแสง

R คือเวกเตอร์ที่ชี้ไปยังทิศทางที่แสงสะท้อนออกจากวัตถุ

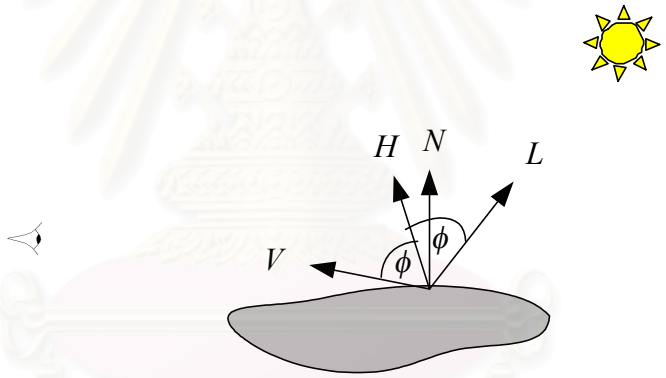
n คือค่าดัชนีที่ใช้จำลองความเรียบมันของวัตถุ

V คือเวกเตอร์ที่ชี้ไปยังผู้สังเกต

Ω คือมุมที่เวกเตอร์ R กระทำกับเวกเตอร์ V

θ คือมุมที่เวกเตอร์ N กระทำกับเวกเตอร์ L เท่ากับมุมที่เวกเตอร์ N กระทำกับเวกเตอร์ R

เนื่องจากการหาเวกเตอร์ R หรือเวกเตอร์ที่ชี้ไปยังทิศทางที่แสงสะท้อนออกจากวัตถุนั้น ต้องใช้การคำนวณค่อนข้างมากซึ่งจะส่งผลต่อประสิทธิภาพในการให้แสงเงาได้ Blinn [4] จึงได้เสนอให้หลีกเลี่ยงการใช้เวกเตอร์ R โดยใช้เวกเตอร์ที่ชี้ไปยังระหว่างกลางของเวกเตอร์ที่ชี้ไปยังแหล่งกำเนิดแสง (L) และเวกเตอร์ที่ชี้ไปยังผู้สังเกต (V) แทน ซึ่งจะเรียกเวกเตอร์นี้ว่า H ดังรูป 2.25 เวกเตอร์ H หาได้จากสมการที่ 2.17 และเขียนสมการที่ 2.16 ให้อยู่ในรูปเวกเตอร์ H ได้ดังสมการที่ 2.18



รูปที่ 2.26 องค์ประกอบการสะท้อนแสงกล้า

$$H = (L+V)/2 \qquad \text{...(2.17)}$$

$$I_s = k_s I_i (N \cdot H)^n \qquad \text{...(2.18)}$$

เมื่อ H คือเวกเตอร์ที่ระหว่างกลางของเวกเตอร์ L และ V

ϕ คือมุมที่เวกเตอร์ L กระทำกับเวกเตอร์ H เท่ากับมุมที่เวกเตอร์ H กระทำกับเวกเตอร์ V

ดังนั้นในการให้แสงเงาจากสมการที่ 2.13 สามารถเขียนใหม่ได้เป็นสมการที่ 2.19 ซึ่งเกิดจากการรวมการสะท้อนแสงทั้ง 3 แบบ คือ การสะท้อนแสงล้อมรอบ การสะท้อนแสงแพร่ และการสะท้อนแสงกล้า เข้าด้วยกัน โดยการแทนสมการที่ 2.14 2.15 และ 2.19 เข้าไปในสมการที่ 2.13 จะได้

$$I = I_a k_a + I_i (k_d (L \cdot N) + k_s (N \cdot H)^n) \quad \text{...(2.19)}$$

ในกรณีที่เป็นรูปสีจะใช้สมการที่ 2.19 กับแต่ละค่าสีแดง เขียว และน้ำเงิน แต่เนื่องจากการสะท้อนแสงกล้าจะให้สีออกมาเป็นสีขาวในกรณีที่แหล่งกำเนิดแสงเป็นสีขาว ดังนั้นการสะท้อนแสงกล้าจะเหมือนกันทั้งในสีแดง เขียว และน้ำเงิน ดังสมการที่ 2.20 2.21 และ 2.22

$$I_r = I_a k_{ar} + I_i (k_{dr} (L \cdot N) + k_s (N \cdot H)^n) \quad \text{...(2.20)}$$

$$I_g = I_a k_{ag} + I_i (k_{dg} (L \cdot N) + k_s (N \cdot H)^n) \quad \text{...(2.21)}$$

$$I_b = I_a k_{ab} + I_i (k_{db} (L \cdot N) + k_s (N \cdot H)^n) \quad \text{...(2.22)}$$

เมื่อ I_r คือค่าความเข้มแสงสีแดงที่วัตถุสะท้อนออกมา

I_g คือค่าความเข้มแสงสีเขียวที่วัตถุสะท้อนออกมา

I_b คือค่าความเข้มแสงสีน้ำเงินที่วัตถุสะท้อนออกมา

k_{ar} คือสัมประสิทธิ์การสะท้อนแสงล้อมรอบสีแดงของวัตถุ

k_{ag} คือสัมประสิทธิ์การสะท้อนแสงล้อมรอบสีเขียวของวัตถุ

k_{ab} คือสัมประสิทธิ์การสะท้อนแสงล้อมรอบสีน้ำเงินของวัตถุ

k_{dr} คือสัมประสิทธิ์การสะท้อนแสงแพร่สีแดงของวัตถุ

k_{dg} คือสัมประสิทธิ์การสะท้อนแสงแพร่สีเขียวของวัตถุ

k_{db} คือสัมประสิทธิ์การสะท้อนแสงแพร่สีน้ำเงินของวัตถุ

2.5 การหาเวกเตอร์ปกติของข้อมูลเชิงปริมาตร

เวกเตอร์ปกติเป็นสิ่งจำเป็นในการให้แสงเงาแก่วัตถุ แต่สำหรับการสร้างภาพเชิงปริมาตรแล้วการคำนวณหาเวกเตอร์ปกติโดยตรงนั้นเป็นสิ่งที่ทำได้ยาก เนื่องจากพื้นผิวในชุดข้อมูลมีตำแหน่งไม่ชัดเจน ดังนั้นจึงใช้เกรเดียนต์ของค่าจุดปริมาตรมาใช้แทนเวกเตอร์ปกติของพื้นผิว การประมาณค่าเกรเดียนต์ของข้อมูลเชิงปริมาตรนั้นสามารถกระทำได้หลายวิธีแต่วิธีที่นิยม ได้แก่

2.5.1 การประมาณค่าเกรเดียนต์แบบเซ็นทรัลดิฟเฟอเรนซ์ (Central Difference Gradient Estimator)

การประมาณค่าแบบนี้เป็นวิธีที่ง่ายและเร็วที่สุด เป็นค่าหาค่าความแตกต่างระหว่างข้อมูลก่อนหน้าและข้อมูลถัดไปของจุดที่ต้องการประมาณค่าตามแค่ละแนวแกน ซึ่งสามารถหาเขียนเป็นสมการได้เป็น

$$g_x = f(x+1,y,z) - f(x-1,y,z) \quad \text{...(2.23)}$$

$$g_y = f(x,y+1,z) - f(x,y-1,z) \quad \text{...(2.24)}$$

$$g_z = f(x,y,z+1) - f(x,y,z-1) \quad \text{...(2.25)}$$

เมื่อ g_x คือค่าเกรเดียนต์ของค่าจุดปริมาตรในแนวแกน x

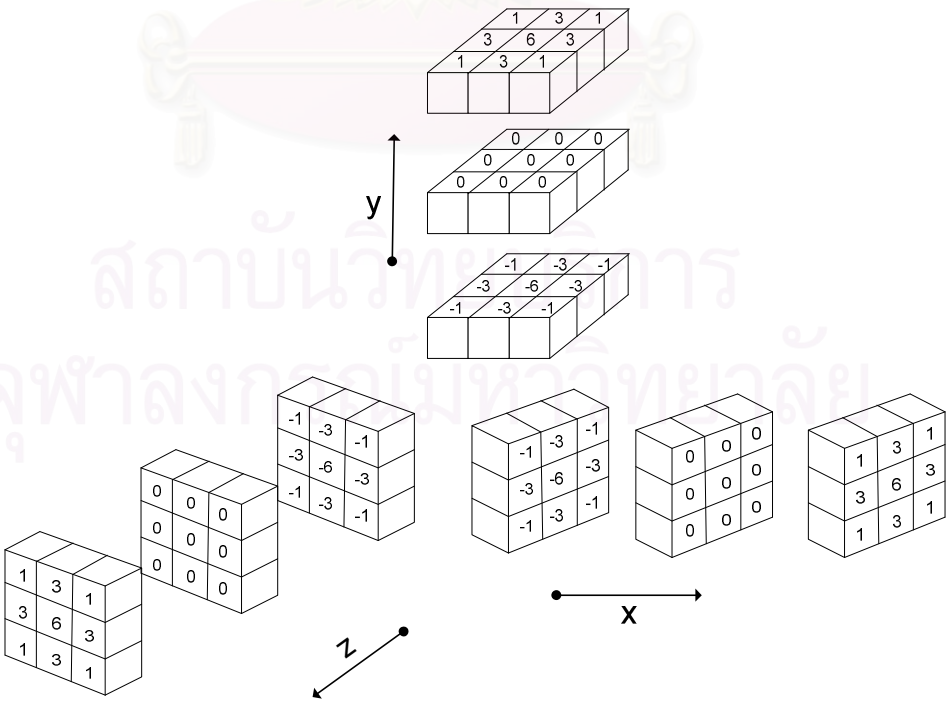
g_y คือค่าเกรเดียนต์ของค่าจุดปริมาตรในแนวแกน y

g_z คือค่าเกรเดียนต์ของค่าจุดปริมาตรในแนวแกน z

$f(x,y,z)$ คือค่าของจุดปริมาตร ณ ตำแหน่ง (x,y,z)

2.5.2 **การประมาณค่าเกรเดียนต์แบบโซเบล** (Sobel Gradient Estimator) [12]

การประมาณค่าแบบนี้ให้คุณภาพของค่าเกรเดียนต์ที่ดีกว่าแบบเซ็นทรัลดิฟเฟอเรนซ์ โดยใช้ค่าจากจุดปริมาตรที่อยู่ล้อมรอบขนาด 3x3 จุดปริมาตร รูปที่ 2.27 แสดงหน้ากากของตัวดำเนินการโซเบลที่ใช้ในการประมาณค่าเกรเดียนต์ของข้อมูลเชิงปริมาตรในแต่ละแกน



รูปที่ 2.27 ตัวดำเนินการโซเบลแบบ 3 มิติ

2.5.3 **การประมาณค่าเกรเดียนต์แบบซูคเคอร์-ฮัมเมล** (Zucker-Hummel Gradient Estimator) [13]

การประมาณค่าแบบนี้ใช้ค่าของจุดปริมาตรข้างเคียงขนาด 3x3 ในการประมาณค่าเกรเดียนต์เช่นเดียวกับการประมาณค่าแบบโซเบล สมการที่ 2.26 2.27 และ 2.28 แสดงการประมาณค่าเกรเดียนต์แบบซูคเคอร์-ฮัมเมลในแต่ละแกน

$$\begin{aligned} g_x = {} & f(x+1,y,z)-f(x-1,y,z)+ \\ & k_1\big(f(x+1,y+1,z+1)-f(x-1,y+1,z+1)+f(x+1,y-1,z+1)-f(x-1,y-1,z+1)+ \\ & \quad f(x+1,y+1,z-1)-f(x-1,y+1,z-1)+f(x+1,y-1,z-1)-f(x-1,y-1,z-1)+ \\ & k_2\big(f(x+1,y,z+1)-f(x-1,y,z+1)+f(x+1,y+1,z+1)-f(x-1,y+1,z+1)+ \\ & \quad f(x+1,y,z-1)-f(x-1,y,z-1)+f(x+1,y-1,z)-f(x-1,y-1,z)\big) \end{aligned} \quad \text{...(2.26)}$$

$$\begin{aligned} g_y = {} & f(x,y+1,z)-f(x,y-1,z)+ \\ & k_1\big(f(x-1,y+1,z-1)-f(x-1,y-1,z+1)+f(x+1,y-1,z+1)-f(x+1,y-1,z+1)+ \\ & \quad f(x-1,y+1,z-1)-f(x-1,y-1,z-1)+f(x+1,y+1,z-1)-f(x+1,y-1,z-1)+ \\ & k_2\big(f(x,y+1,z+1)-f(x-1,y,z+1)+f(x+1,y+1,z)-f(x+1,y+1,z)+ \\ & \quad f(x,y+1,z-1)-f(x,y-1,z-1)+f(x+1,y-1,z)-f(x+1,y-1,z)\big) \end{aligned} \quad \text{...(2.27)}$$

$$\begin{aligned} g_z = {} & f(x,y,z+1)-f(x,y,z-1)+ \\ & k_1\big(f(x-1,y+1,z+1)-f(x-1,y+1,z-1)+f(x+1,y+1,z+1)-f(x+1,y+1,z-1)+ \\ & \quad f(x-1,y-1,z+1)-f(x-1,y-1,z-1)+f(x+1,y-1,z+1)-f(x+1,y-1,z-1)+ \\ & k_2\big(f(x,y+1,z+1)-f(x,y+1,z+1)+f(x,y-1,z+1)-f(x,y-1,z-1)+ \\ & \quad f(x+1,y,z+1)-f(x+1,y,z-1)+f(x-1,y,z+1)-f(x-1,y,z-1)\big) \end{aligned} \quad \text{...(2.28)}$$

เมื่อ $k_1 = \dfrac{\sqrt{3}}{3}$ และ $k_2 = \dfrac{\sqrt{2}}{2}$

เมื่อนำองค์ประกอบแต่ละแนวแกนของค่าเกรเดียนต์ของจุดปริมาตรมารวมกันจะได้เวกเตอร์ของเกรเดียนต์ในสามมิติ แต่เนื่องจากเวกเตอร์ปกติของพื้นผิวต้องเป็นเวกเตอร์หนึ่งหน่วยดังนั้นต้องทำให้เป็นเวกเตอร์หนึ่งหน่วย (normalize) ด้วยสมการที่ 2.29 ก็จะได้เวกเตอร์ปกติของพื้นผิวตามต้องการ

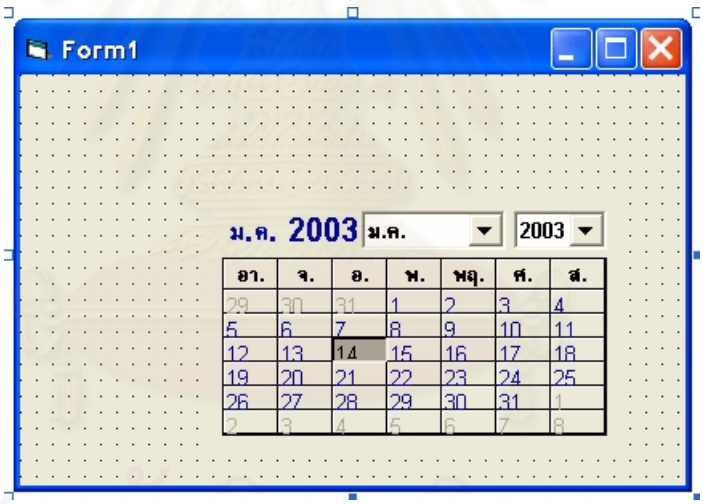
$$N = g/|g| \quad (2.29)$$

เมื่อ N คือเวกเตอร์ปกติของพื้นผิว

g คือเวกเตอร์เกรเดียนต์ของความหนาแน่นของปริมาตร

2.6 แอ็กทิฟเอกซ์คอนโทรล (ActiveX control) [14]

แอ็กทิฟเอกซ์คอนโทรลเป็นแอ็กทิฟเอกซ์คอมโพเนนท์ที่ทำหน้าที่ติดต่อกับผู้ใช้ โดยพัฒนามาจากโอเล (OLE - Object Linking and Embedding) คอนโทรล คุณสมบัติของแอ็กทิฟเอกซ์คอนโทรลคือต้องเป็นวัตถุคอม (COM-Component Object Model) และสนับสนุนการลงทะเบียนด้วยตัวเอง (Self-Registration) การทำงานของแอ็กทิฟเอกซ์คอนโทรลจะเป็นการทำงานในแบบกระบวนการเดียวกัน (In-Process Execution) เท่านั้น ในการใช้งานแอ็กทิฟเอกซ์คอนโทรลจะถูกนำไปฝังกับโปรแกรมประยุกต์อื่นโดยจะเรียกโปรแกรมประยุกต์ที่นำแอ็กทิฟเอกซ์คอนโทรลไปฝังว่าคอนโทรลคอนเทนเนอร์ (control container) ดังตัวอย่างรูปที่ 2.28 แสดงให้เห็นว่า Visual Basic ฟอร์มจะเป็นคอนโทรลคอนเทนเนอร์ของแอ็กทิฟเอกซ์คอนโทรลที่เป็นปฏิทิน โปรแกรมที่สามารถเป็นคอนโทรลคอนเทนเนอร์นี้มีเป็นจำนวนมาก โปรแกรมที่ใช้งานทั่วไป เช่น Microsoft Word, Microsoft Excel, Internet Explorer รวมถึงเครื่องมือที่ใช้พัฒนาโปรแกรม เช่น Visual Basic, Visual C++, Delphi ก็สามารถเป็นคอนโทรลคอนเทนเนอร์ได้เพียงแค่สนับสนุนเทคโนโลยีแอ็กทิฟเอกซ์



รูปที่ 2.28 แอ็กทิฟเอกซ์คอนโทรลและคอนโทรลคอนเทนเนอร์

การสื่อสารระหว่างแอ็กทิฟเอกซ์คอนโทรลและคอนโทรลคอนเทนเนอร์สามารถทำได้ 3 วิธี คือ

- พร็อพเพอร์ตี้ (Properties) เป็นคุณสมบัติของแอ็กทิฟเอกซ์คอนโทรลที่เห็นและแก้ไขได้โดยคอนโทรลคอนเทนเนอร์
- เมธอด (Method) คล้ายกับฟังก์ชันในแอ็กทิฟเอกซ์คอนโทรลที่สามารถถูกเรียกใช้ได้โดยคอนโทรลคอนเทนเนอร์ โดยคอนโทรลคอนเทนเนอร์สามารถส่งค่าตัวแปรผ่านเมธอดให้แอ็กทิฟเอกซ์คอนโทรลได้
- อีเว็นต์ (Event) เป็นข้อความที่ถูกส่งจากแอ็กทิฟเอกซ์คอนโทรลไปยังคอนโทรลคอนเทนเนอร์เพื่อแจ้งให้ทราบว่ามีเหตุการณ์เกิดขึ้น

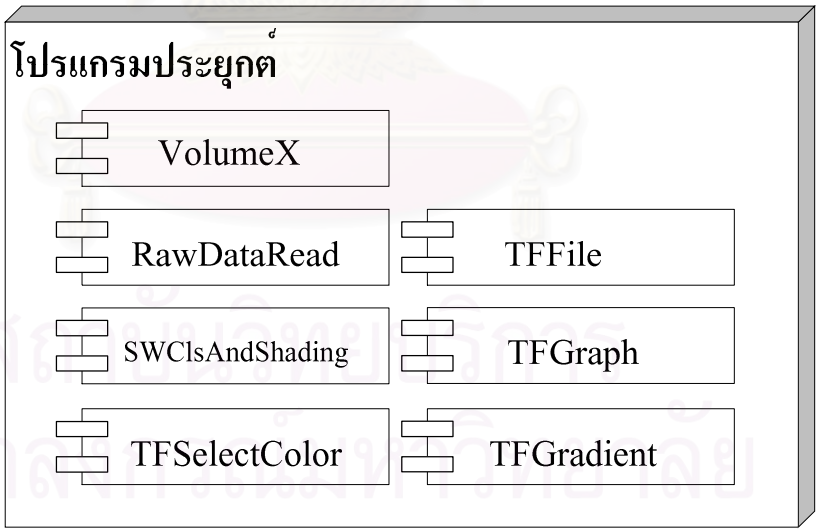
บทที่ 3

การออกแบบระบบ

ในบทนี้จะกล่าวถึงรายละเอียดในการออกแบบระบบชุดส่วนโปรแกรมที่ใช้ในการสร้างภาพเชิงปริมาตร เนื้อหาจะประกอบไปด้วยภาพรวมของระบบ และการออกแบบส่วนโปรแกรมแต่ละส่วน ซึ่งมีรายละเอียดดังนี้

3.1 ภาพรวมของระบบ

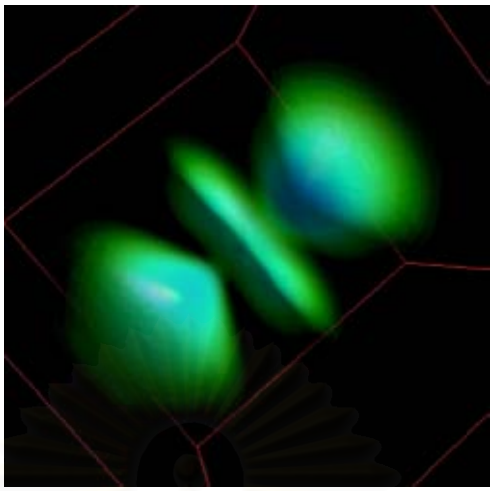
ระบบของชุดส่วนโปรแกรมที่ใช้ในการสร้างภาพเชิงปริมาตรจะประกอบไปด้วยชุดส่วนโปรแกรมจำนวน 7 ส่วน โดยส่วนโปรแกรมทั้งหมดจะอยู่ในรูปของแอ็กทิฟเอกซ์คอนโทรล โดยมีโปรแกรมประยุกต์ที่ผู้ใช้สร้างขึ้นเป็นคอนโทรลคอนเทนเนอร์ดังรูปที่ 3.1 โดยแต่ละส่วนโปรแกรมจะทำงานเป็นอิสระต่อกัน แต่ข้อมูลนำเข้าและส่งออกของแต่ละส่วนโปรแกรมอยู่ในรูปแบบเดียวกัน โดยโปรแกรมประยุกต์สามารถกำหนดการส่งผ่านข้อมูลและการทำงานร่วมกันของส่วนโปรแกรมแต่ละตัวได้



รูปที่ 3.1 ภาพรวมของระบบชุดส่วนโปรแกรมในการสร้างภาพเชิงปริมาตร

3.2 ส่วนโปรแกรม VolumeX

ส่วนโปรแกรม VolumeX เป็นส่วนโปรแกรมหลักของงานวิจัยมีหน้าที่ในการสร้างภาพเชิงปริมาตรจากข้อมูลนำเข้า โดยส่วนโปรแกรมนี้จะมีส่วนต่อประสานกับผู้ใช้เป็นพื้นที่ในการแสดงผลภาพผลลัพธ์ที่ได้จากการสร้างภาพเชิงปริมาตรดังรูปที่ 3.2



รูปที่ 3.2 ภาพแสดงส่วนติดต่อกับผู้ใช้ของส่วนโปรแกรม VolumeX

ส่วนโปรแกรม VolumeX จะมีการทำงาน 5 ภาวะ ได้แก่

ภาวะ 0 เดินเครื่องเปล่า ในภาวะนี้ส่วนโปรแกรมจะไม่ทำการสร้างภาพของข้อมูลนำเข้า จะแสดงผลเป็นสีพื้นหลังและข้อความที่ผู้ใช้ต้องการ

ภาวะ 1 การสร้างภาพโดยตรง ภาวะการทำงานนี้จะใช้เมื่อข้อมูลนำเข้าอยู่ในรูปแบบของค่าสีและความทึบแสง (RGBA) ส่วนโปรแกรมจะทำการสร้างภาพจากข้อมูลนี้ตามขั้นตอนวิธีการสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์ 3 มิติโดยไม่ทำการประมวลผลใด ๆ กับข้อมูลนำเข้า

ภาวะ 2 การสร้างภาพปกติ ในภาวะนี้ส่วนโปรแกรมจะได้รับข้อมูลนำเข้าในรูปแบบของข้อมูลเชิงปริมาตรที่เป็นค่าสเกลาร์และค่าฉลาก หรือเฉพาะค่าใดค่าหนึ่ง รวมกับฟังก์ชันถ่ายโอน ส่วนโปรแกรมจะสร้างภาพจากข้อมูลนำเข้านี้โดยใช้วิธีการสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์ 3 มิติร่วมกับการทำการจำแนกประเภทโดยใช้ความสามารถในการทำแฟรกเม็นต์เชดดิงแบบโปรแกรมได้ของกราฟิกส์ฮาร์ดแวร์ (รายละเอียดการทำงานในบทที่ 4)

ภาวะ 3 การสร้างภาพปกติ + การสะท้อนแสงแพร่ ภาวะการทำงานนี้ส่วนโปรแกรมมีการทำงานเหมือนกับในภาวะการสร้างภาพปกติ แต่เพิ่มการให้แสงเงาแบบการสะท้อนแสงแพร่ในการสร้างภาพ ซึ่งจะเพิ่มการทำงานในส่วนของการหาเวกเตอร์ปกติของพื้นผิวจากข้อมูลสเกลาร์ (บทที่ 2.5) และการให้แสงเงาโดยโดยใช้ความสามารถในการทำแฟรกเม็นต์เชดดิงแบบโปรแกรมได้ของกราฟิกส์ฮาร์ดแวร์ (รายละเอียดการทำงานในบทที่ 4)

ภาวะ 4 การสร้างภาพปกติ + การสะท้อนแสงแพร่ + การสะท้อนแสงกล้า ในภาวะการทำงานนี้ส่วนโปรแกรมจะทำงานเหมือนในภาวะ 3 เพียงแต่เพิ่มการให้แสงเงาแบบการสะท้อนแสงกล้า

ส่วนต่อประสานของส่วนโปรแกรม VolumeX นี้ประกอบไปด้วย

1) คุณสมบัติ 13 คุณสมบัติ (property) ดังตารางที่ 3.1

ตารางที่ 3.1 คุณสมบัติของส่วนโปรแกรม VolumeX

ชื่อคุณสมบัติ	ชนิด	ค่าเริ่มต้น	ความหมาย
AlphaNumber	float	0.3	ค่าที่ใช้ในการทดสอบค่าความทึบแสงของแฟรกเม็นต์
AlphaTestMode	short integer	0	0 – แฟรกเม็นต์ที่มีค่าความทึบแสงมากกว่าค่า AlphaNumber จะถูกเก็บไว้ 1 – แฟรกเม็นต์ที่มีค่าความทึบแสงน้อยกว่าค่า AlphaNumber จะถูกเก็บไว้ 2 – แฟรกเม็นต์ที่มีค่าความทึบแสงเท่ากับค่า AlphaNumber จะถูกเก็บไว้เท่ากับ 3 – แฟรกเม็นต์จะถูกเก็บไว้ทั้งหมด
BlendingMode	short integer	0	0 – การรวมค่าสีโดยใช้ค่าความทึบแสงถ่วงน้ำหนัก (อัลฟาเบล็นดิง) 1 – การรวมค่าสีโดยใช้ค่าสีที่มีค่าความทึบแสงน้อยที่สุด 2 – การรวมค่าสีโดยใช้ค่าสีที่มีความทึบแสงมากที่สุด 3 – ไม่ทำการรวมค่าสี
ContinuousRendering	bool	false	กำหนดให้ทำการสร้างภาพอย่างต่อเนื่อง
EnableMouseHandle	bool	true	กำหนดให้ส่วนโปรแกรมจัดการกับเหตุการณ์ที่เกิดขึ้นจากเมาส์
GradientEstMethod	short integer	2	0 – ใช้การประมาณค่าเกรเดียนต์แบบเซ็นทรัลดิฟเฟอเรนซ์ 1 – ใช้การประมาณค่าเกรเดียนต์แบบโซเบล 2 – ใช้การประมาณค่าเกรเดียนต์แบบซูคเคอร์-ฮัมเมล
InvertClipGeometry	bool	false	true – กำหนดให้การตัดแบบใช้รูปทรงให้ตัดส่วนที่อยู่ภายในรูปทรงออก false – กำหนดให้การตัดแบบใช้รูปทรงให้ตัดส่วนที่อยู่ภายนอกรูปทรงออก

ชื่อคุณสมบัติ	ชนิด	ค่าเริ่มต้น	ความหมาย
PopUpError	bool	true	กำหนดให้มีการแสดงผลเมื่อมีความผิดพลาดในการทำงาน
ShowBoundaryBox	bool	true	กำหนดให้แสดงขอบเขตของข้อมูลเชิงปริมาตร
ShowFPS	bool	false	กำหนดให้มีการแสดงจำนวนภาพที่สร้างได้ต่อวินาที
Slices	short integer	128	กำหนดจำนวนรีแซมพลิงสไลซ์
TextureFilter	short integer	0	กำหนดตัวกรองที่ใช้ในการรีแซมพลิง 0 – ตัวกรองแบบเต็นท์ (การประมาณค่าในช่วงแบบเชิงเส้น) 1 – ตัวกรองแบบกล่อง (การประมาณค่าในช่วงแบบเพื่อนบ้านที่ใกล้ที่สุด)
Zoom	float	1.0	กำหนดอัตราการขยายภาพ

2) **เมท็อด** 29 เมท็อด(method) ซึ่งมีรายละเอียดดังต่อไปนี้

CaptureToBMP	
รายละเอียด	ใช้จัดเก็บภาพที่สร้างลงใน BMP ไฟล์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	BSTR FileName – ชื่อไฟล์ที่ต้องการจัดเก็บ

CaptureToClipboard	
รายละเอียด	ใช้จัดเก็บภาพที่สร้างลงในคลิปบอร์ด
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

CheckCapability	
รายละเอียด	ใช้ทดสอบความสามารถของเครื่องว่าสามารถใช้ภาวะในการสร้างภาพได้หรือไม่
ค่าส่งกลับ	BOOL – True สามารถใช้ได้ False ไม่สามารถใช้ได้
พารามิเตอร์	short integer – ภาวะการสร้างภาพ

DeleteClipGeometry	
รายละเอียด	ใช้ลบรูปทรงที่ใช้ในการตัดออกจากบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

DeletePrimaryData	
รายละเอียด	ใช้ลบข้อมูลหลักออกจากบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

DeleteSecondaryData	
รายละเอียด	ใช้ลบข้อมูลรองออกจากบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

DeleteTransferFunc	
รายละเอียด	ใช้ลบฟังก์ชันถ่ายโอนออกจากบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

DisableClipGeometry	
รายละเอียด	ใช้ปิดการตัดโดยใช้รูปทรง
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

DisableClipGeometry	
รายละเอียด	ใช้ปิดการตัดโดยใช้ระนาบ
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

EditGradient	
รายละเอียด	ใช้แก้ไขข้อมูลเกรเดียนต์เวกเตอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – OffsetX ตำแหน่งในแกน x
พารามิเตอร์	short integer – OffsetY ตำแหน่งในแกน y
พารามิเตอร์	short integer – OffsetZ ตำแหน่งในแกน z
พารามิเตอร์	byte array – แถวลำดับของข้อมูลเกรเดียนต์เวกเตอร์

EditPrimaryData	
รายละเอียด	ใช้แก้ไขข้อมูลหลักในบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – OffsetX ตำแหน่งในแกน x
พารามิเตอร์	short integer – OffsetY ตำแหน่งในแกน y
พารามิเตอร์	short integer – OffsetZ ตำแหน่งในแกน z
พารามิเตอร์	byte array – แถวลำดับของข้อมูลหลัก

EditSecondaryData	
รายละเอียด	ใช้แก้ไขข้อมูลข้อมูลรอง
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – OffsetX ตำแหน่งในแกน x
พารามิเตอร์	short integer – OffsetY ตำแหน่งในแกน y
พารามิเตอร์	short integer – OffsetZ ตำแหน่งในแกน z
พารามิเตอร์	byte array – แถวลำดับของข้อมูลรอง

EditTransferFunc	
รายละเอียด	ใช้แก้ไขข้อมูลเกรเดียนต์เวกเตอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – OffsetX ตำแหน่งในแกน x
พารามิเตอร์	short integer – OffsetY ตำแหน่งในแกน y
พารามิเตอร์	byte array – แถวลำดับของข้อมูลฟังก์ชันถ่ายโอน

EnableClipGeometry	
รายละเอียด	ใช้เปิดความสามารถในการตัดโดยใช้รูปทรง
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	float – PosX ตำแหน่งในแกน x
พารามิเตอร์	float – PosY ตำแหน่งในแกน y
พารามิเตอร์	float – PosZ ตำแหน่งในแกน z
พารามิเตอร์	float – UpX ขนาดของเวกเตอร์ที่ใช้กำหนดทิศทางการวางรูปทรงที่ใช้ตัดในแนวแกน x

พารามิเตอร์	float – UpY ขนาดของเวกเตอร์ที่ใช้กำหนดทิศทางการวางรูปทรงที่ใช้ตัดในแนวแกน y
พารามิเตอร์	float – UpZ ขนาดของเวกเตอร์ที่ใช้กำหนดทิศทางการวางรูปทรงที่ใช้ตัดในแนวแกน z

EnableClipPlane	
รายละเอียด	ใช้เปิดความสามารถในการตัดโดยใช้ระนาบ
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	float – A ค่าสัมประสิทธิ์ของ x ในสมการระนาบ
พารามิเตอร์	float – B ค่าสัมประสิทธิ์ของ y ในสมการระนาบ
พารามิเตอร์	float – C ค่าสัมประสิทธิ์ของ z ในสมการระนาบ
พารามิเตอร์	float – D ค่าคงที่ในสมการระนาบ

GetErrorString	
รายละเอียด	ใช้เรียกข้อความซึ่งอธิบายรหัสระบุความผิดพลาด
ค่าส่งกลับ	BSTR – ข้อความอธิบายรหัสความผิดพลาด
พารามิเตอร์	short integer – ErrorCode รหัสความผิดพลาด

LoadClipGeometry	
รายละเอียด	ใช้บรรจุข้อมูลของรูปทรงที่จะได้ใช้ลงในบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	float array – VertexBuffer แถวลำดับของจุดยอด
พารามิเตอร์	integer array – FaceBuffer แถวลำดับของหน้ารูปทรง

LoadPrimaryData	
รายละเอียด	ใช้บรรจุข้อมูลหลักลงในบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	byte array – Buffer แถวลำดับของข้อมูลหลัก

LoadSecondaryData	
รายละเอียด	ใช้บรรจุข้อมูลรองลงในบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

พารามิเตอร์	byte array – Buffer แถวลำดับของข้อมูลรอง

LoadTransferFunc	
รายละเอียด	ใช้บรรจุข้อมูลของฟังก์ชันถ่ายโอนลงในบัฟเฟอร์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	byte array – Buffer แถวลำดับของข้อมูลฟังก์ชันถ่ายโอน

PrintText	
รายละเอียด	ใช้พิมพ์ข้อความลงบนภาพผลลัพธ์
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – Number หมายเลขของข้อความ 0-9
พารามิเตอร์	short integer – X ตำแหน่งในแนวแกน x
พารามิเตอร์	short integer – Y ตำแหน่งในแนวแกน y
พารามิเตอร์	short integer – Red ค่าความเข้มของสีแดง
พารามิเตอร์	short integer – Green ค่าความเข้มของสีเขียว
พารามิเตอร์	short integer – Blue ค่าความเข้มของสีน้ำเงิน
พารามิเตอร์	BSTR – Text ข้อความ

ResetView	
รายละเอียด	ใช้ตั้งมุมมองของวัตถุให้กลับไปยังตำแหน่งเริ่มต้น
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด

Rotate	
รายละเอียด	ใช้หมุนมุมมองวัตถุ
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	float – Angle องศาที่ต้องการหมุน
พารามิเตอร์	float – X ค่าเวกเตอร์ของแกนหมุนในแนวแกน x
พารามิเตอร์	float – Y ค่าเวกเตอร์ของแกนหมุนในแนวแกน y
พารามิเตอร์	float – Z ค่าเวกเตอร์ของแกนหมุนในแนวแกน z

SetAmbientColor	
รายละเอียด	ใช้ตั้งค่าแสงแวดล้อม
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – Red ค่าความเข้มของแสงสีแดง
พารามิเตอร์	short integer – Green ค่าความเข้มของแสงสีเขียว
พารามิเตอร์	short integer – Blue ค่าความเข้มของแสงสีน้ำเงิน

SetBackgroundColor	
รายละเอียด	ใช้ตั้งค่าสีพื้นหลัง
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – Red ค่าความเข้มของสีแดง
พารามิเตอร์	short integer – Green ค่าความเข้มของสีเขียว
พารามิเตอร์	short integer – Blue ค่าความเข้มของสีน้ำเงิน

SetBoundaryBoxColor	
รายละเอียด	ใช้ตั้งค่าสีกล่องที่ใช้แสดงขอบเขตข้อมูล
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – Red ค่าความเข้มของสีแดง
พารามิเตอร์	short integer – Green ค่าความเข้มของสีเขียว
พารามิเตอร์	short integer – Blue ค่าความเข้มของสีน้ำเงิน

SetLightDirection	
รายละเอียด	ใช้ตั้งค่าทิศทางที่แสงส่อง
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	float – X ค่าเวกเตอร์ของทิศทางของแสงในแนวแกน x
พารามิเตอร์	float – Y ค่าเวกเตอร์ของทิศทางของแสงในแนวแกน y
พารามิเตอร์	float – Z ค่าเวกเตอร์ของทิศทางของแสงในแนวแกน z

SetRenderingMode	
รายละเอียด	ใช้ตั้งค่าภาวะการสร้างภาพ
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	short integer – Mode ภาวะการสร้างภาพ

SetScale	
รายละเอียด	ใช้ตั้งมาตราส่วนของข้อมูล
ค่าส่งกลับ	short integer – รหัสระบุความผิดพลาด
พารามิเตอร์	float – X ค่ามาตราส่วนในแนวแกน x
พารามิเตอร์	float – Y ค่ามาตราส่วนในแนวแกน y
พารามิเตอร์	float – Z ค่ามาตราส่วนในแนวแกน z

3.3 ส่วนโปรแกรม RawDataRead

ส่วนโปรแกรม RawDataRead เป็นส่วนโปรแกรมที่มีความสามารถในการอ่านข้อมูลจากแฟ้มข้อมูลดิบ โดยสามารถอ่านได้ทั้งข้อมูลเชิงปริมาตร และข้อมูลฟังก์ชันถ่ายโอน นอกจากนั้นยังมีสามารถอ่านข้อมูลแถวลำดับของจุดยอด และแถวลำดับของหน้ารูปทรง จากแฟ้มส่งออกฉากแบบแอสกี (ASCII Scene Export) ของโปรแกรม 3D Studio Max เพื่อใช้ในการตัดโดยใช้รูปทรง ส่วนโปรแกรมนี้จะไม่มีส่วนต่อประสานกับผู้ใช้

ส่วนต่อประสานของส่วนโปรแกรมนี้ประกอบไปด้วย 4 เมท็อด ได้แก่

FacesArrRead	
รายละเอียด	ใช้อ่านข้อมูลหน้ารูปทรงจากแฟ้มส่งออกฉากแบบแอสกี
ค่าส่งกลับ	integer array – แถวลำดับของข้อมูลจุดยอด
พารามิเตอร์	string – FileName ชื่อแฟ้มข้อมูลที่ต้องการอ่าน

RawRead	
รายละเอียด	ใช้อ่านข้อมูลเชิงปริมาตรจากแฟ้มข้อมูลดิบ
ค่าส่งกลับ	byte array – แถวลำดับของข้อมูลเชิงปริมาตร
พารามิเตอร์	string – FileName ชื่อแฟ้มข้อมูลที่ต้องการอ่าน
พารามิเตอร์	long integer – SizeX ขนาดของข้อมูลเชิงปริมาตรในแกน x
พารามิเตอร์	long integer – SizeY ขนาดของข้อมูลเชิงปริมาตรในแกน y
พารามิเตอร์	long integer – SizeZ ขนาดของข้อมูลเชิงปริมาตรในแกน z

TFRead	
รายละเอียด	ใช้อ่านข้อมูลฟังก์ชันถ่ายโอนจากแฟ้มข้อมูลดิบ
ค่าส่งกลับ	byte array – แถวลำดับของข้อมูลฟังก์ชันถ่ายโอน
พารามิเตอร์	string – FileName ชื่อแฟ้มข้อมูลที่ต้องการอ่าน

VertexArrRead	
รายละเอียด	ใช้อ่านข้อมูลจุดยอดจากแฟ้มส่งออกฉากแบบแอสกี
ค่าส่งกลับ	float array – แถวลำดับของข้อมูลจุดยอด
พารามิเตอร์	string – FileName ชื่อแฟ้มข้อมูลที่ต้องการอ่าน

3.4 ส่วนโปรแกรม TFFile

ส่วนโปรแกรม TFFile นี้สร้างขึ้นมาเพื่อจัดเก็บและอ่านแฟ้มข้อมูลที่ใช้เก็บฟังก์ชันถ่ายโอน โดยรูปแบบของแฟ้มนั้นจะเป็นรูปแบบเฉพาะที่สร้างขึ้นเนื่องจากยังไม่มีรูปแบบแฟ้มข้อมูลมาตรฐานที่แพร่หลายสำหรับเก็บฟังก์ชันถ่ายโอน ส่วนโปรแกรมนี้ไม่มีส่วนต่อประสานกับผู้ใช้

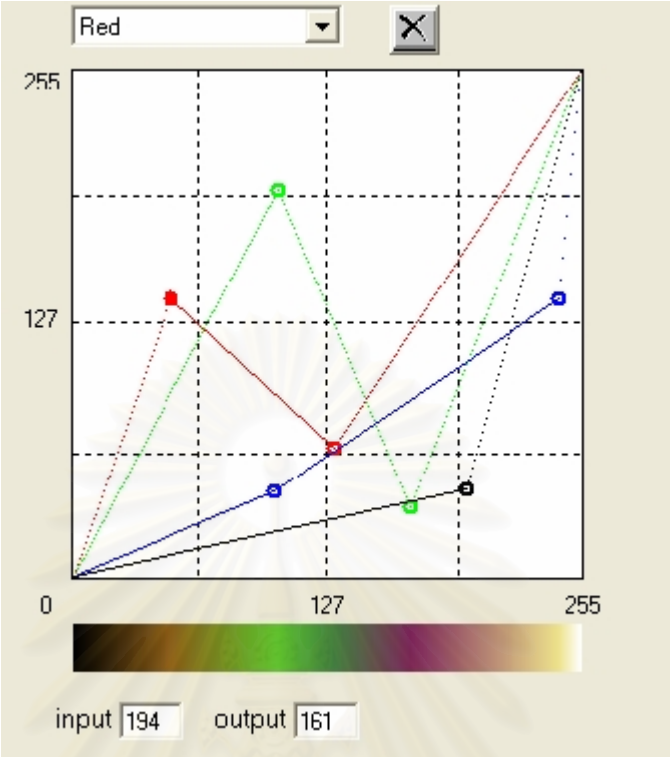
ส่วนต่อประสานของส่วนโปรแกรม TFFile ประกอบไปด้วย 2 เมท็อด ได้แก่

LoadTF	
รายละเอียด	ใช้อ่านแฟ้มข้อมูลฟังก์ชันถ่ายโอน
ค่าส่งกลับ	byte array – แถวลำดับของข้อมูลฟังก์ชันถ่ายโอน
พารามิเตอร์	string – FileName ชื่อแฟ้มข้อมูลที่ต้องการอ่าน

SaveTF	
รายละเอียด	ใช้จัดเก็บข้อมูลฟังก์ชันถ่ายโอนลงในแฟ้มข้อมูล
ค่าส่งกลับ	-
พารามิเตอร์	string – FileName ชื่อแฟ้มข้อมูลที่ต้องการจัดเก็บ
พารามิเตอร์	byte array – แถวลำดับของข้อมูลฟังก์ชันถ่ายโอนที่ต้องการจัดเก็บ

3.5 ส่วนโปรแกรม TFGraph

ส่วนโปรแกรม TFGraph เป็นส่วนโปรแกรมที่สร้างขึ้นเพื่ออำนวยความสะดวกให้กับผู้ใช้ในการสร้างฟังก์ชันถ่ายโอนที่เหมาะสมให้กับข้อมูลเชิงปริมาตรที่มีลักษณะต่อเนื่อง หรือข้อมูลเชิงปริมาตรของค่าสเกลาร์ โดยส่วนโปรแกรมนี้จะสร้างฟังก์ชันถ่ายโอนจากกราฟซึ่งผู้ใช้วาดจากการกำหนดจุดควบคุม (control point) และนำเส้นกราฟลากผ่านจุดควบคุมไปเป็นฟังก์ชันถ่ายโอน ส่วนต่อประสานกับผู้ใช้ของส่วนโปรแกรมนี้แสดงในรูปที่ 3.3 ผู้ใช้สามารถเลือกที่จะสร้างกราฟของแต่ละสี (แดง เขียว น้ำเงิน และความทึบแสง) ได้จากรายการด้านบน และคลิกเมาส์เพื่อกำหนดจุดควบคุมของสีนั้น แกนนอนของกราฟเป็นสเกลของค่าสเกลาร์นำเข้าและแกนตั้งเป็นค่าความเข้มของสีส่งออก แถบสีด้านล่างแสดงค่าสีจริงเมื่อทำการผสมค่าสีแดง เขียว และน้ำเงิน ของกราฟที่ตำแหน่งนั้นนั้น



รูปที่ 3.3 ส่วนต่อประสานกับผู้ใช้ของส่วนโปรแกรม TFGraph

ส่วนต่อประสานของส่วนโปรแกรม TFGraph นี้จะประกอบด้วย 2 เมท็อด ได้แก่

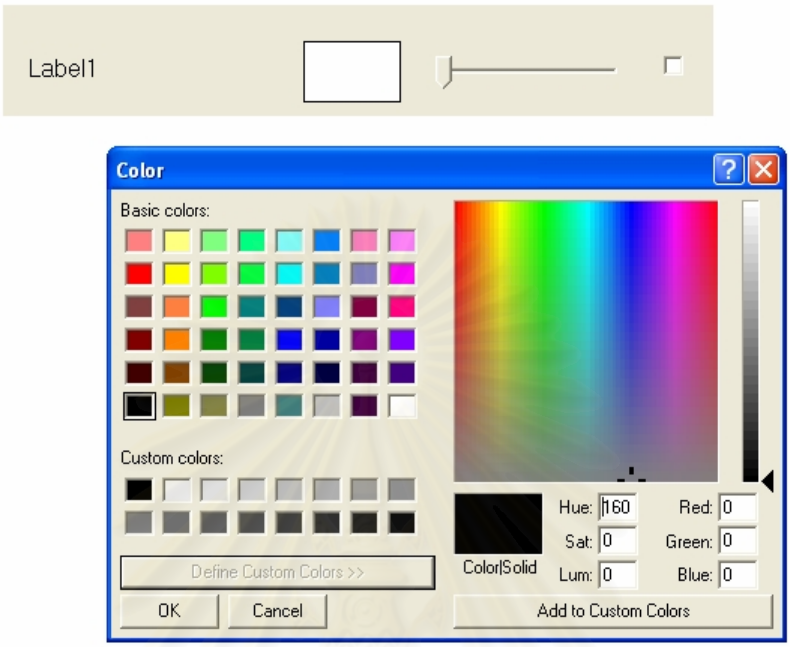
GetTF	
รายละเอียด	ใช้อ่านค่าฟังก์ชันถ่ายโอนจากส่วนโปรแกรม
ค่าส่งกลับ	byte array – ข้อมูลฟังก์ชันถ่ายโอน

Reset	
รายละเอียด	ใช้กำหนดค่าฟังก์ชันถ่ายโอนไปยังค่าเริ่มต้น
ค่าส่งกลับ	-

3.6 ส่วนโปรแกรม TFSelectColor

ส่วนโปรแกรม TFSelectColor สร้างขึ้นเพื่ออำนวยความสะดวกในการสร้างฟังก์ชันถ่ายโอนให้กับข้อมูลเชิงปริมาตรที่มีลักษณะไม่ต่อเนื่อง หรือข้อมูลเชิงปริมาตรของค่าฉลากที่ได้จากการแบ่งส่วน ส่วนโปรแกรมนี้จะให้ผู้ใช้สร้างฟังก์ชันถ่ายโอนจากการเลือกสี และความทึบแสงให้กับค่าฉลากแต่ละค่า ส่วนต่อประสานกับผู้ใช้ของส่วนโปรแกรมนี้แสดงในรูป 3.4 เมื่อผู้ใช้คลิกที่กล่องสีส่วนโปรแกรมจะแสดงจานสีขึ้นมาให้ผู้ใช้เลือกเลือกสีที่ต้องการให้กับค่าฉลากนั้น ผู้ใช้กำหนดค่าความทึบแสงได้โดยเลื่อนสไลด์บาร์ (slide bar) ด้านข้างของกล่องสี และกล่องเลือกด้านขวาใช้สำหรับกำหนดให้แสดงหรือไม่แสดงภาพ

ของค่าฉลากนั้น ในการสร้างฟังก์ชันถ่ายโอนของข้อมูลเชิงปริมาตรซึ่งมีค่าฉลากหลายค่าต้องใช้ส่วนโปรแกรมนี้เป็นจำนวนเท่ากับค่าฉลากทั้งหมดที่มีในข้อมูลจึงจะสามารถกำหนดค่าสีให้กับทุกค่าฉลากได้



รูปที่ 3.4 ส่วนต่อประสานกับผู้ใช้ของส่วนโปรแกรม TFSelectColor

ส่วนต่อประสานของส่วนโปรแกรมนี้จะประกอบไปด้วย

1) **คุณสมบัติ** ตามตารางที่ 3.2

ตารางที่ 3.2 คุณสมบัติของส่วนโปรแกรม TFSelecColor

ชื่อคุณสมบัติ	ชนิด	ค่าเริ่มต้น	ความหมาย
Caption	string	Label1	เป็นข้อความที่แสดงให้ผู้ใช้รู้ความหมายของค่าฉลาก

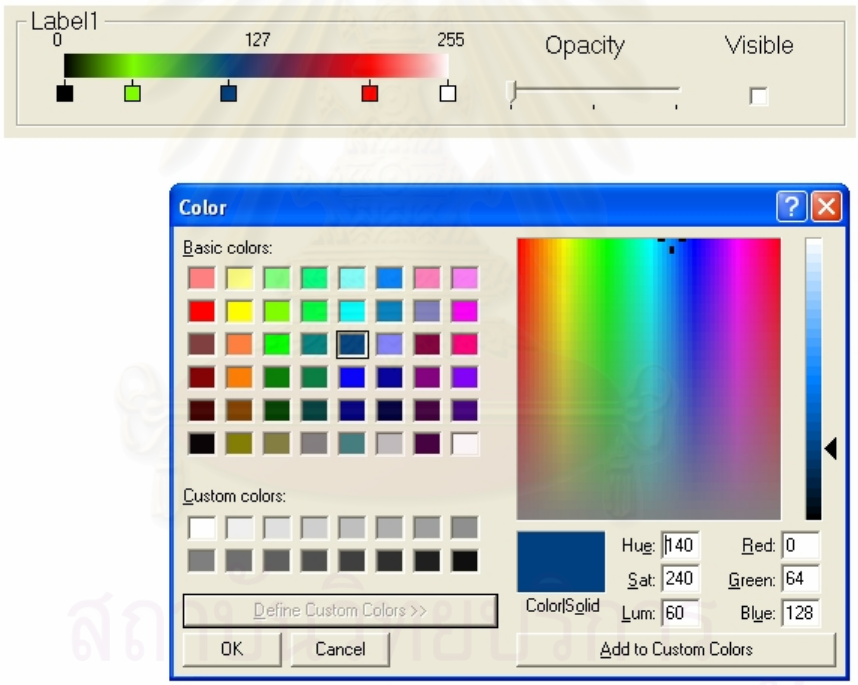
2) **เมท็อด** 2 เมท็อด ได้แก่

GetTF	
รายละเอียด	ใช้อ่านค่าฟังก์ชันถ่ายโอนจากส่วนโปรแกรม
ค่าส่งกลับ	byte array – ข้อมูลฟังก์ชันถ่ายโอน

Reset	
รายละเอียด	ใช้กำหนดค่าฟังก์ชันถ่ายโอนไปยังค่าเริ่มต้น
ค่าส่งกลับ	-

3.7 ส่วนโปรแกรม TFGradient

ส่วนโปรแกรม TFGradient สร้างขึ้นเพื่ออำนวยความสะดวกให้กับผู้ใช้ในการสร้างฟังก์ชันถ่ายโอนให้กับการสร้างภาพเชิงปริมาตรโดยใช้ข้อมูลทั้งแบบต่อเนื่องและไม่ต่อเนื่องร่วมกัน ส่วนโปรแกรมนี้จะให้ผู้ใช้สร้างฟังก์ชันถ่ายโอนจากการเลือกโทนสี และความทึบแสงให้กับค่าฉลากแต่ละค่า ส่วนต่อประสานกับผู้ใช้ของส่วนโปรแกรมนี้แสดงในรูป 3.5 เมื่อผู้ใช้คลิกที่ด้านล่างของแถบสีส่วนโปรแกรมจะสร้างจุดควบคุมขึ้นที่ตำแหน่งนั้น และเมื่อผู้ใช้คลิกขวาที่จุดควบคุมส่วนโปรแกรมจะแสดงจานสีขึ้นมาให้ผู้ใช้เลือกสีที่ต้องการให้กับจุดควบคุมนั้น โทนสีบนแถบสีจะถูกสร้างขึ้นจากการไล่สีตามตำแหน่งและสีของจุดควบคุม ผู้ใช้กำหนดค่าความทึบแสงได้โดยเลื่อนสไลด์บาร์ (slide bar) ด้านข้างของแถบสี และกล่องเลือกด้านขวาใช้สำหรับกำหนดให้แสดงหรือไม่แสดงภาพของค่าฉลากนั้น ในการสร้างฟังก์ชันถ่ายโอนของข้อมูลเชิงปริมาตรซึ่งมีค่าฉลากหลายค่าต้องใช้ส่วนโปรแกรมนี้เป็นจำนวนเท่ากับค่าฉลากทั้งหมดที่มีในข้อมูลจึงจะสามารถกำหนดค่าสีให้กับทุกค่าฉลากได้



รูปที่ 3.5 ส่วนต่อประสานกับผู้ใช้ของส่วนโปรแกรม TFGradient

ส่วนต่อประสานของส่วนโปรแกรมนี้จะประกอบไปด้วย

1) **คุณสมบัติ** ตามตารางที่ 3.3

ตารางที่ 3.3 คุณสมบัติของส่วนโปรแกรม TFGradient

ชื่อคุณสมบัติ	ชนิด	ค่าเริ่มต้น	ความหมาย
Caption	string	Label1	เป็นข้อความที่แสดงให้ผู้ใช้รู้ความหมายของค่าฉลาก

2) **เมท็อด** 2 เมท็อด ได้แก่

GetTF	
รายละเอียด	ใช้อ่านค่าฟังก์ชันถ่ายโอนจากส่วนโปรแกรม
ค่าส่งกลับ	byte array – ข้อมูลฟังก์ชันถ่ายโอน

Reset	
รายละเอียด	ใช้กำหนดค่าฟังก์ชันถ่ายโอนไปยังค่าเริ่มต้น
ค่าส่งกลับ	-

3.8 ส่วนโปรแกรม SWClsAndShading

ส่วนโปรแกรม SWClsAndShading สร้างขึ้นเพื่อใช้ในกรณีที่ผู้ใช้ไม่ต้องการใช้กราฟิกส์ฮาร์ดแวร์ในการทำการจำแนกประเภทและการให้แสงเงา ข้อมูลนำเข้าของส่วนโปรแกรมนี้จะประกอบไปด้วยข้อมูลเชิงปริมาตรของค่าสเกลาร์และค่าฉลาก ฟังก์ชันถ่ายโอน ทิศทางของแสง และความเข้มของแสงแวดล้อม ส่วนโปรแกรมจะทำการจำแนกประเภทและให้แสงเงาโดยใช้ซอฟต์แวร์จากนั้นจะได้ข้อมูลส่งออกในรูปของข้อมูลเชิงปริมาตรของค่าสีและความทึบแสง ซึ่งสามารถนำไปสร้างภาพได้โดยตรงโดยใช้ภาวะการทำงานที่ 1 ของส่วนโปรแกรม VolumeX

ส่วนโปรแกรมนี้ไม่มีส่วนต่อประสานกับผู้ใช้ และประกอบด้วย 1 เมท็อด คือ

SWClsAndShading	
รายละเอียด	ใช้คำนวณการจำแนกประเภทและแสงเงาโดยใช้ซอฟต์แวร์
ค่าส่งกลับ	byte array – ข้อมูลเชิงปริมาตรของค่าสีและความทึบแสง
พารามิเตอร์	Primary - byte array ข้อมูลเชิงปริมาตรหลัก
พารามิเตอร์	Secondary – byte array ข้อมูลเชิงปริมาตรรอง
พารามิเตอร์	TF – byte array ข้อมูลฟังก์ชันถ่ายโอน
พารามิเตอร์	LightDir – double array ทิศทางของแสง
พารามิเตอร์	AmbColor – byte array ค่าความเข้มของแสงแวดล้อม

บทที่ 4

ขั้นตอนวิธีในการสร้างภาพเชิงปริมาตร

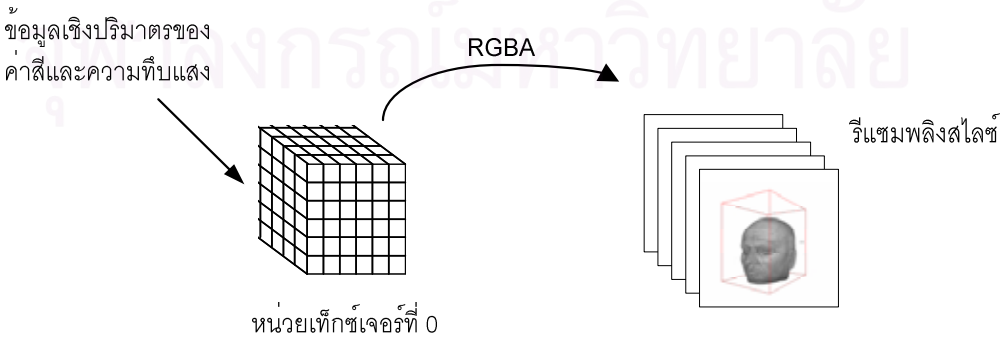
ในบทนี้จะกล่าวถึงรายละเอียดของขั้นตอนวิธีในการสร้างภาพเชิงปริมาตรที่ใช้ในส่วนโปรแกรมหลักของงานวิจัย โดยจะแบ่งเป็นหัวข้อดังต่อไปนี้ ขั้นตอนวิธีในการสร้างภาพโดยตรง ขั้นตอนวิธีในการสร้างภาพปกติ ขั้นตอนวิธีในการให้แสงเงาแบบแสงแพร่ ขั้นตอนวิธีในการให้แสงเงาแบบแสงกล้า และขั้นตอนวิธีในการตัด ซึ่งมีรายละเอียดในแต่ละหัวข้อดังนี้

4.1 ขั้นตอนวิธีในการสร้างภาพโดยตรง

ขั้นตอนวิธีในการสร้างภาพโดยตรงนี้ถูกใช้ใน ภาวะ 1 ของส่วนโปรแกรม VolumeX โดยข้อมูลนำเข้าจะอยู่ในรูปแบบของข้อมูลเชิงปริมาตรของค่าสีและความทึบแสงซึ่งสามารถนำไปสร้างภาพได้โดยตรงตามขั้นตอนวิธีการสร้างภาพเชิงปริมาตรโดยใช้การแม็พเท็กซ์เจอร์สามมิติ ซึ่งจะมีขั้นตอนการทำงานดังต่อไปนี้

1. แปลงข้อมูลเชิงปริมาตรของค่าสีและความทึบแสงให้เป็นเท็กซ์เจอร์ 3 มิติ เนื่องจากทั้งเท็กซ์เจอร์ 3 มิติและข้อมูลเชิงปริมาตรถูกจัดเก็บในรูปแบบแถวลำดับเหมือนกันการแปลงจึงสามารถทำได้โดยตรง
2. ยึดเหนี่ยว (Bind) เท็กซ์เจอร์ที่ได้เข้ากับหน่วยเท็กซ์เจอร์ที่ 0
3. แม็พหน่วยเท็กซ์เจอร์ที่ 0 เข้ากับรีแซมพลิงสไลซ์

ภาพรวมขั้นตอนการทำงานแสดงในรูปที่ 4.1 เมื่อกราฟิกส์ฮาร์ดแวร์วาดรีแซมพลิงสไลซ์ค่าสีและความทึบแสงจะถูกอ่านมาจากหน่วยเท็กซ์เจอร์ที่ 0 ที่ถูกแม็พอยู่กับรีแซมพลิงสไลซ์



รูปที่ 4.1 ขั้นตอนการสร้างภาพโดยตรง

4.2 ขั้นตอนวิธีในการสร้างภาพปกติ

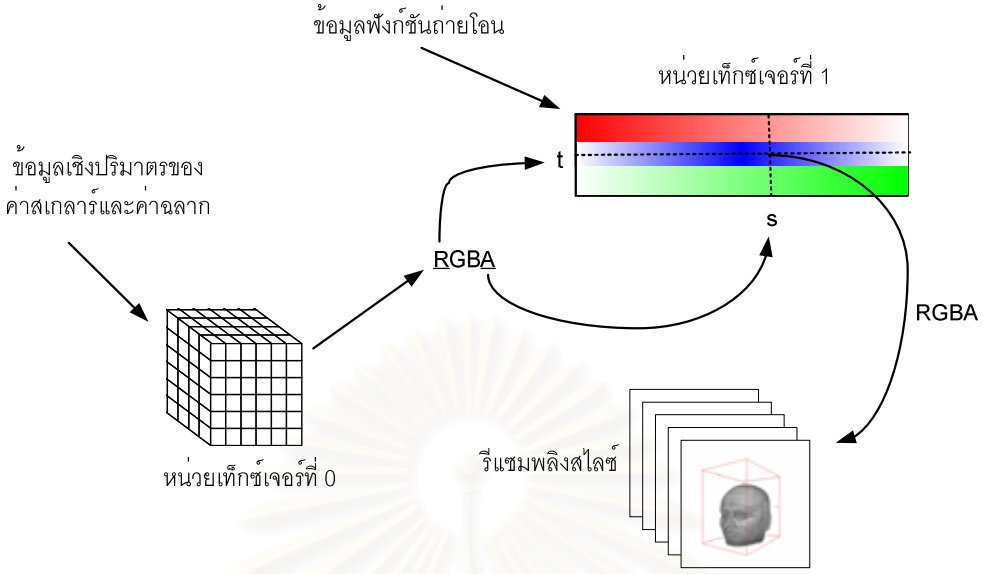
ขั้นตอนวิธีในการสร้างภาพโดยตรงนี้ถูกใช้ใน ภาวะ 2 ของส่วนโปรแกรม VolumeX โดยข้อมูลนำเข้าจะประกอบไปด้วยข้อมูลเชิงปริมาตรของค่าสเกลาร์และข้อมูลเชิงปริมาตรของค่าฉลาก หรือเพียงอย่างใดอย่างหนึ่งเท่านั้น รวมกับข้อมูลฟังก์ชันถ่ายโอน ในรายละเอียดขั้นตอนวิธีนี้จะกล่าวถึงขั้นตอนการใช้ทั้งข้อมูลสเกลาร์และฉลากร่วมกันเนื่องจากการใช้เพียงอย่างใดอย่างหนึ่งจะแตกต่างกันเพียงปิดความสามารถนั้น ในการสร้างภาพจากข้อมูลนำเข้านี้จะแตกต่างจากการสร้างภาพโดยตรงคือก่อนที่จะสร้างภาพต้องมีการจำแนกประเภทเพื่อแปลงค่าสเกลาร์และค่าฉลากให้กลายเป็นค่าสีและความทึบแสงก่อน ในขั้นตอนวิธีนี้จะใช้ความสามารถการอ่านเท็กซ์เจอร์แบบดีเพ็นแดนต์เท็กซ์เจอร์ริงของกราฟิกส์ฮาร์ดแวร์ในการประยุกต์การจำแนกประเภทแบบหลัง เนื่องจากการใช้ความสามารถการอ่านเท็กซ์เจอร์แบบดีเพ็นแดนต์เท็กซ์เจอร์ริงในกราฟิกส์ฮาร์ดแวร์ของเอ็นวิเดียร์และเอทีไอมีขั้นตอนไม่เหมือนกันดังนั้นจึงแบ่งออกเป็น 2 หัวข้อ ได้แก่

4.2.1 ขั้นตอนวิธีในการสร้างภาพปกติบนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์

บนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์การทำดีเพ็นแดนต์เท็กซ์เจอร์ริงต้องใช้ความสามารถของ เท็กซ์เจอร์เชดเดอร์ ซึ่งจะมีขั้นตอนวิธีดังต่อไปนี้

1. แปลงข้อมูลค่าสเกลาร์และข้อมูลค่าฉลากให้เป็นเท็กซ์เจอร์ 3 มิติ โดยให้ค่าสเกลาร์อยู่ในช่องความทึบแสง (A) และข้อมูลค่าฉลากอยู่ในช่องสีแดง (R)
2. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้เข้ากับหน่วยเท็กซ์เจอร์ที่ 0
3. แปลงข้อมูลฟังก์ชันถ่ายโอนให้เป็นเท็กซ์เจอร์
4. ยึดเหนี่ยวเท็กซ์เจอร์ของฟังก์ชันถ่ายโอนเข้ากับหน่วยเท็กซ์เจอร์ที่ 1
5. กำหนดให้เท็กซ์เจอร์เชดเดอร์ของหน่วยเท็กซ์เจอร์ที่ 1 ใช้ตัวดำเนินการ Dependent Alpha-Red Texturing ในการอ่านค่า
6. แม็พหน่วยเท็กซ์เจอร์ที่ 0 และ 1 เข้ากับรีแซมพลิงสไลซ์

ภาพรวมขั้นตอนการทำงานแสดงในรูปที่ 4.2 เมื่อกราฟิกส์ฮาร์ดแวร์วาดรีแซมพลิงสไลซ์จะอ่านค่าจากเท็กซ์เจอร์หน่วยที่ 0 ซึ่งเก็บค่าสเกลาร์อยู่ใน ช่องความทึบแสง (A) และข้อมูลค่าฉลากอยู่ในช่องสีแดง (R) จากนั้นจึงอ่านค่าเท็กซ์เจอร์หน่วยถัดไปคือหน่วยที่ 1 ซึ่งถูกโปรแกรมให้ใช้ตัวดำเนินการ Dependent Alpha-Red Texturing ซึ่งจะส่งผลให้หน่วยเท็กซ์เจอร์ที่ 1 ใช้ค่าของช่องความทึบแสงและช่องสีแดงของหน่วยเท็กซ์เจอร์ที่ 0 ในการกำหนดตำแหน่งพิกัดในการอ่านค่า ซึ่งการทำเช่นนี้เสมือนเป็นการทำการจำแนกประเภท เนื่องจากมีการป้อนค่าสเกลาร์และค่าฉลากของข้อมูลเข้าไปในฟังก์ชันถ่ายโอนแล้วได้ค่าสีและความทึบแสงออกมา ดังนั้นค่าสีและความทึบแสงที่ได้จากหน่วยเท็กซ์เจอร์ที่ 1 จึงเป็นค่าสีและความทึบแสงผลลัพธ์ซึ่งถูกนำไปเขียนลงบนรีแซมพลิงสไลซ์



รูปที่ 4.2 ขั้นตอนการสร้างภาพปกติบนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์

4.2.2 ขั้นตอนวิธีในการสร้างภาพปกติบนเอทีไอกราฟิกส์ฮาร์ดแวร์

บนเอทีไอกราฟิกส์ฮาร์ดแวร์การทำดีเพ็นแดนต์เท็กซ์เจอร์ริงต้องใช้ความสามารถในการทำแฟรกเม็นต์เชดดิง ซึ่งจะมีขั้นตอนวิธีดังต่อไปนี้

1. แปลงข้อมูลค่าสเกลาร์ให้เป็นเท็กซ์เจอร์ 3 มิติ
2. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 0
3. แปลงข้อมูลค่าฉลากให้เป็นเท็กซ์เจอร์ 3 มิติ
4. ยึดเหนี่ยวแท็กเจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 5
5. แปลงข้อมูลฟังก์ชันถ่ายโอนให้เป็นเท็กซ์เจอร์
6. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 1
7. โปรแกรมแฟรกเม็นต์เชดดิงตามรหัสดังรูปที่ 4.3
8. แม็พหน่วยเท็กซ์เจอร์ที่ 0 1 และ 5 เข้ากับรีแซมพลิงสไลซ์

รหัสโปรแกรมแฟรกเม็นต์เชดดิงในรูปที่ 4.3 จะเริ่มจาก

<u>บรรทัดที่ 3</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 0 ซึ่งเก็บค่าสเกลาร์มาเก็บไว้ในเรจิสเตอร์ 0

<u>บรรทัดที่ 4</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 5 ซึ่งเก็บค่าฉลากมาเก็บไว้ในเรจิสเตอร์ 5

<u>บรรทัดที่ 6-8</u> ย้ายค่าสีแดงของเรจิสเตอร์ 5 มาเก็บที่เรจิสเตอร์ 2 ในทุกช่องสี

บรรทัดที่ 10-12 ย้ายค่าสีในช่องความทึบแสงของเรจิสเตอร์ 0 มาเก็บไว้ในช่องสีแดงของเรจิสเตอร์ 2 ดังนั้นขณะนี้ค่าของเรจิสเตอร์ 2 จะประกอบไปด้วยค่าสเกลาร์ในช่องสีแดงและค่าฉลากในช่องสีเขียวและน้ำเงิน

จบรอบการทำงานแรก

บรรทัดที่ 16 อ่านค่าหน่วยเท็กซ์เจอร์ 1 ซึ่งเก็บฟังก์ชันถ่ายโอนโดยใช้ค่าในเรจิสเตอร์ 2 ในการกำหนดตำแหน่งพิกัดในการอ่าน ทำให้เกิดการอ่านค่าแบบดีเพ็นแดนต์ ดังนั้นค่าที่ได้ในเรจิสเตอร์ 1 จึงเป็นค่าสีและความทึบแสงผลลัพธ์

บรรทัดที่ 19-25 ย้ายค่าสีและความทึบแสงจากเรจิสเตอร์ 1 ไปเก็บ ไว้ในเรจิสเตอร์ 0 ซึ่งเป็นเรจิสเตอร์ส่งออก ค่าที่ได้จึงถูกนำไปเขียนลงบนรีแซมพลิงสไลซ์

```
1   glBeginFragmentShaderATI();
2
3     glSampleMapATI(GL_REG_0_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Pri. Volume Data
4     glSampleMapATI(GL_REG_5_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Sec. Volume Data
5
6     glColorFragmentOp1ATI(GL_MOV_ATI,
7               GL_REG_2_ATI, GL_NONE, GL_NONE,
8               GL_REG_5_ATI, GL_RED, GL_NONE);
9
10    glColorFragmentOp1ATI(GL_MOV_ATI,
11              GL_REG_2_ATI, GL_RED_BIT_ATI, GL_NONE,
12              GL_REG_0_ATI, GL_ALPHA, GL_NONE);
13
14
15    //End of first pass
16    glSampleMapATI(GL_REG_1_ATI, GL_REG_2_ATI,GL_SWIZZLE_STR_ATI); //Sample TF
17
18
19    glColorFragmentOp1ATI(GL_MOV_ATI,
20              GL_REG_0_ATI, GL_NONE, GL_NONE,
21              GL_REG_1_ATI, GL_NONE, GL_NONE);
22
23    glAlphaFragmentOp1ATI(GL_MOV_ATI,
24              GL_REG_0_ATI, GL_NONE,
25              GL_REG_1_ATI, GL_ALPHA, GL_NONE);
26
27
28   glEndFragmentShaderATI();
```

รูปที่ 4.3 รหัสโปรแกรมแฟรกเม็นต์เชดดิงในการสร้างภาพปกติ

4.3 ขั้นตอนวิธีในการให้แสงเงาแบบแสงแพร่

ขั้นตอนวิธีในการสร้างภาพโดยตรงนี้ถูกใช้ใน ภาวะ 3 ของส่วนโปรแกรม VolumeX โดยข้อมูลนำเข้าจะเหมือนกับในขั้นตอนวิธีการสร้างภาพปกติ เพียงเพิ่มทิศทางของแสงเข้ามา จากสมการที่ 2.19 ตัดเอาพจน์ของการสะท้อนแสงกล้าออกไป และให้สัมประสิทธิ์ของการสะท้อนแสงแวดล้อมและแสงแพร่เท่ากันคือสีของวัตถุ จะสามารถเขียนสมการใหม่ได้เป็น

$$I = base \times \left(I_a + N \bullet L\right) \qquad \ldots(4.1)$$

เมื่อ I คือความเข้มของแสงที่วัตถุสะท้อนออกมา

$base$ คือค่าสีของพื้นผิววัตถุ

I_a คือความเข้มของแสงแวดล้อม

N คือเวกเตอร์ปกติของพื้นผิว

L คือทิศทางของแสง

จากสมการที่ 4.1 ในการคำนวณการให้แสงเงาแบบแสงแพร่นั้นต้องใช้เวกเตอร์ปกติของพื้นผิววัตถุซึ่งสามารถหาได้จากการใช้เกรเดียนต์เวกเตอร์ (หัวข้อ 2.5) การจะใช้กราฟิกส์ฮาร์ดแวร์ในการคำนวณการให้แสงเงาจึงต้องมีการส่งค่าเวกเตอร์ปกติของข้อมูลเชิงปริมาตรเข้าไปยังกราฟิกส์ฮาร์ดแวร์ ซึ่งสามารถทำได้โดยการเก็บเวกเตอร์ปกติลงในเท็กซ์เจอร์ 3 มิติที่มีขนาดเท่ากับเท็กซ์เจอร์ของข้อมูลเชิงปริมาตร โดยการแปลงขนาดของเวกเตอร์ตามแนวแกน x y และ z ให้เป็นค่าความเข้มของสีตามช่องแดง เขียว และน้ำเงิน ตามลำดับ ซึ่งการแปลงนี้สามารถทำได้โดยใช้สมการที่ 4.2 4.3 และ 4.4

$$R = \frac{N_x + 1}{2} \times 255 \qquad \text{...(4.2)}$$

$$G = \frac{N_y + 1}{2} \times 255 \qquad \text{...(4.3)}$$

$$B = \frac{N_z + 1}{2} \times 255 \qquad \text{...(4.4)}$$

เมื่อ R คือค่าความเข้มของสีแดง

G คือค่าความเข้มของสีเขียว

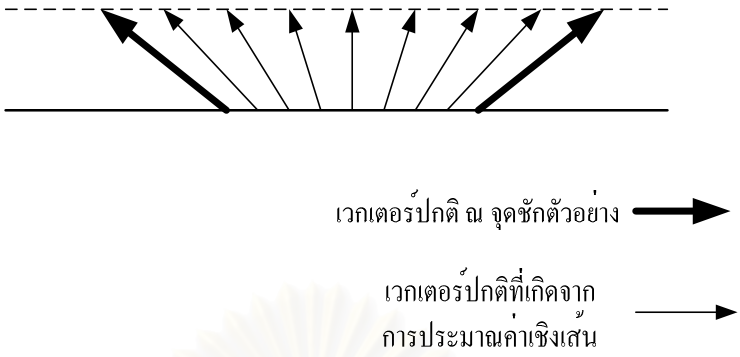
B คือค่าความเข้มของสีน้ำเงิน

N_x คือขนาดของเวกเตอร์ปกติบนแกน x

N_y คือขนาดของเวกเตอร์ปกติบนแกน y

N_z คือขนาดของเวกเตอร์ปกติบนแกน z

แม้ว่าการแปลงเวกเตอร์ปกติให้กลายเป็นเท็กซ์เจอร์จะช่วยให้ส่งเวกเตอร์ปกติไปยังกราฟิกส์ฮาร์ดแวร์ได้แต่ในขณะที่กราฟิกส์ฮาร์ดแวร์อ่านข้อมูลจากเท็กซ์เจอร์นั้นกราฟิกส์ฮาร์ดแวร์จะอาศัยการประมาณค่าเชิงเส้นในการประกอบให้คืนสภาพเพื่อชักตัวอย่างออกจากเท็กซ์เจอร์ ซึ่งการประมาณค่าเชิงเส้นนี้อาจจะส่งผลให้เวกเตอร์ปกติที่ได้ไม่เป็นเวกเตอร์หนึ่งหน่วย ดังรูปที่ 4.4 ขนาดของเวกเตอร์ปกติที่ได้จะไม่คงที่ การคำนวณการให้แสงเงาโดยที่เวกเตอร์ปกติไม่เป็นเวกเตอร์หนึ่งหน่วยจะส่งผลให้ค่าที่ได้มีค่าผิดเพี้ยนจากที่ควรจะเป็น



รูปที่ 4.4 เวกเตอร์ปกติที่เกิดจากการประมาณค่าเชิงเส้น

การแก้เวกเตอร์ปกติที่ไม่เป็นเวกเตอร์หนึ่งหน่วยนี้สามารถทำได้โดยใช้การอ่านเท็กซ์เจอร์แบบดีเพ็นแดนต์กับเท็กซ์เจอร์ชนิดคิวบ์แม็พ โดยนำค่าเวกเตอร์ปกติที่ได้ไปเป็นค่าตำแหน่งในการอ่านเท็กซ์เจอร์คิวบ์แม็พซึ่งแต่ละด้านของลูกบาศก์เก็บค่าเวกเตอร์หนึ่งหน่วยซึ่งชี้ไปที่ตำแหน่งนั้นนั้น เมื่อกราฟิกส์ฮาร์ดแวร์อ่านค่าคิวบ์แม็พ ณ ตำแหน่งที่เวกเตอร์ชี้ไปผลที่ได้คือค่าเวกเตอร์ซึ่งมีทิศทางเดียวกับเวกเตอร์ปกติแต่เป็นเวกเตอร์หนึ่งหน่วย ซึ่งจะเรียกคิวบ์แม็พแบบนี้ว่า นอร์มอลไลเซชันคิวบ์แม็พ (Normalization Cube Map)

เช่นเดียวกับการสร้างภาพปกติขั้นตอนวิธีการทำงานสามารถแบ่งเป็น 2 หัวข้อ ได้แก่

4.3.1 ขั้นตอนวิธีในการให้แสงเงาแบบแสงแพร่บนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์

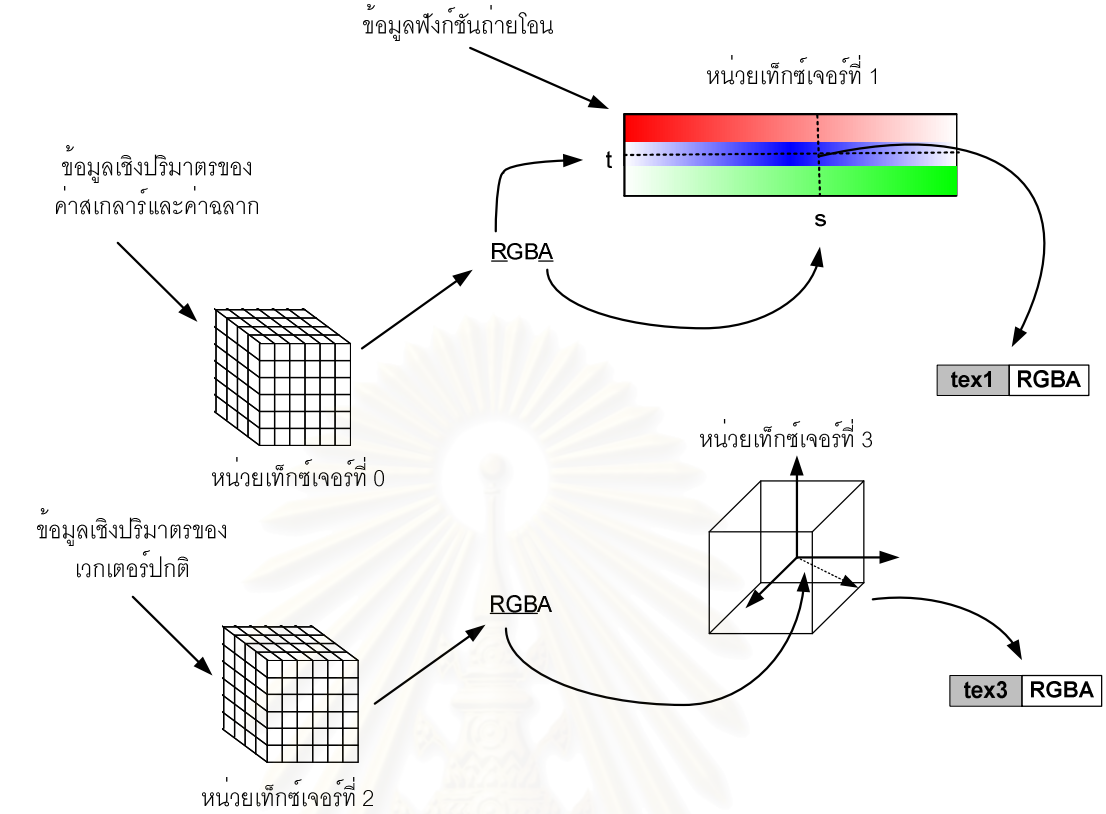
บนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์ขั้นตอนวิธีในการให้แสงเงาแบบแสงแพร่ต้องใช้ความสามารถของเท็กซ์เจอร์เชดเดอร์ร่วมกับเรจิสเตอร์คอมไบเนอร์ ซึ่งจะมีขั้นตอนวิธีดังต่อไปนี้

1. แปลงข้อมูลค่าสเกลาร์และข้อมูลค่าฉลากให้เป็นเท็กซ์เจอร์ 3 มิติ โดยให้ค่าสเกลาร์อยู่ในช่องความทึบแสง (A) และข้อมูลค่าฉลากอยู่ในช่องสีแดง (R)
2. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้เข้ากับหน่วยเท็กซ์เจอร์ที่ 0
3. แปลงข้อมูลฟังก์ชันถ่ายโอนให้เป็นเท็กซ์เจอร์
4. ยึดเหนี่ยวเท็กซ์เจอร์ของฟังก์ชันถ่ายโอนเข้ากับหน่วยเท็กซ์เจอร์ที่ 1
5. คำนวณเวกเตอร์ปกติของข้อมูลเชิงปริมาตร
6. สร้างเท็กซ์เจอร์ 3 มิติเพื่อเก็บเวกเตอร์ปกติ
7. ยึดเหนี่ยวเท็กซ์เจอร์ของเวกเตอร์ปกติเข้ากับหน่วยเท็กซ์เจอร์ที่ 2
8. สร้างนอร์มอลไลเซชันคิวบ์แม็พ
9. ยึดเหนี่ยวนอร์มอลไลเซชันคิวบ์แม็พเข้ากับหน่วยเท็กซ์เจอร์ที่ 3
10. แปลงเวกเตอร์ทิศทางของแสงเป็นค่าสีแล้วเก็บไว้ในเรจิสเตอร์คงตัว 0 (const0)
11. เก็บค่าความเข้มของแสงแวดล้อมลงในเรจิสเตอร์สีรอง (col1)
12. กำหนดให้เท็กซ์เจอร์เชดเดอร์ของหน่วยเท็กซ์เจอร์ที่ 1 ใช้ตัวดำเนินการ Dependent

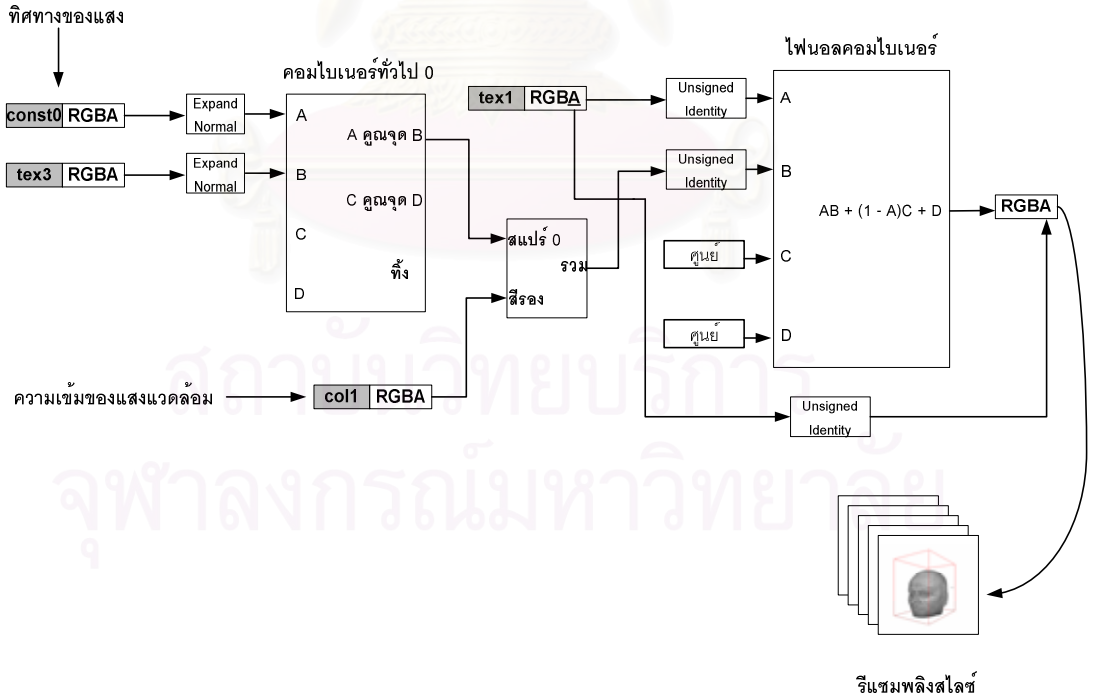
Alpha-Red Texturing ในการอ่านค่า

13. กำหนดให้เท็กซ์เจอร์เซดเดอร์ของหน่วยเท็กซ์เจอร์ที่ 3 ใช้ตัวดำเนินการ Dependent RGB Texuture Cube Map
14. โปรแกรมเรจิสเตอร์คอมไบเนอร์ให้คำนวณตามสมการที่ 4.1 ดังรูปที่ 4.5 ข
15. แม็พหน่วยเท็กซ์เจอร์ที่ 0 1 2 และ 3 เข้ากับรีแซมพลิงสไลซ์

ภาพรวมการทำงานของระบบแสดงดังรูปที่ 4.5 การอ่านค่าของหน่วยเท็กซ์เจอร์ที่ 0 และ 1 จะเหมือนกับในขั้นตอนการสร้างภาพปกติ ในหน่วยเท็กซ์เจอร์ที่ 2 จะเป็นการอ่านค่าเวกเตอร์ปกติจากนั้นนำไปนอร์มอลไลซ์โดยใช้การอ่านเท็กซ์เจอร์แบบดีเพ็นแดนซ์กับเท็กซ์เจอร์คิวแม็พที่อยู่ในหน่วยเท็กซ์เจอร์ที่ 3 จบขั้นตอนการทำงานของเท็กซ์เจอร์เซดเดอร์ค่าที่อ่านได้จากหน่วยเท็กซ์เจอร์ที่ 0 1 2 และ 3 จะถูกเก็บอยู่ในเรจิสเตอร์ tex0 tex1 tex2 และ tex3 ตามลำดับ แล้วส่งการทำงานให้ส่วนเรจิสเตอร์คอมไบเนอร์ต่อไป ในส่วนนี้ค่าเวกเตอร์ปกติในเรจิสเตอร์ tex3 และทิศทางของแสงในเรจิสเตอร์ const0 จะถูกป้อนให้กับคอมไบเนอร์ทั่วไปที่ 0 ในช่อง A และ B ตามลำดับโดยขณะเข้าจะผ่านอินพุตแม็พพิงแบบ Expand Normal เพื่อเปลี่ยนค่าความเข้มของสีแดง เขียว และน้ำเงิน ให้กลับเป็นขนาดของเวกเตอร์ในแนวแกน x y และ z ตามลำดับ โปรแกรมให้คอมไบเนอร์ทั่วไปที่ 0 ทำการคูณจุดข้อมูลนำเข้าผลลัพธ์ของการคูณจุดจะถูกส่งผ่านให้ไฟนอลคอมไบเนอร์ผ่านทางเรจิสเตอร์สแปร์ 0 เพื่อนำไปรวมกับค่าความเข้มของแสงแวดล้อมที่ถูกเก็บไว้ในเรจิสเตอร์ col1 (สีรอง) ผลของการรวมจะถูกป้อนเข้าที่ช่อง B ของไฟนอลคอมไบเนอร์ ในขณะที่ค่าสีที่ได้จากฟังก์ชันถ่ายโอนซึ่งอยู่ในเรจิสเตอร์ tex1 จะถูกป้อนเข้าที่ช่อง A ไฟนอลคอมไบเนอร์จะคูณค่าสีในช่อง A และ B ได้ค่าสีผลลัพธ์ตามการคำนวณในสมการที่ 4.1 (C และ D มีค่าเป็น 0) ค่าความทึบแสงจะถูกส่งผ่านโดยตรงจากเรจิสเตอร์ tex1 ไปเป็นค่าความทึบแสงของผลลัพธ์โดยไม่มีการคำนวณ จากในขั้นตอนสุดท้ายนั้นค่าสีและความทึบแสงที่ได้จะถูกนำไปเขียนลงบนรีแซมพลิงสไลซ์



(ก) ส่วนเท็กซ์เจอร์เชดเดอร์



(ข) ส่วนเรจิสเตอร์คอมไบเนอร์

รูปที่ 4.5 ขั้นตอนการให้แสงเงาแบบแสงแพร่บนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์

4.3.2 ขั้นตอนวิธีในการให้แสงเงาแบบแสงแพร่บนเอทีไอกราฟิกส์ฮาร์ดแวร์

บนเอทีไอกราฟิกส์ฮาร์ดแวร์ทั้งการให้แสงเงาแบบแสงแพร่และการอ่านเท็กซ์เจอร์แบบดีเพ็นแดนต์ใช้เพียงความสามารถในการทำแฟรกเม็นต์เชดดิง ซึ่งจะมีขั้นตอนวิธีดังต่อไปนี้

1. แปลงข้อมูลค่าสเกลาร์ให้เป็นเท็กซ์เจอร์ 3 มิติ
2. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 0
3. แปลงข้อมูลค่าฉลากให้เป็นเท็กซ์เจอร์ 3 มิติ
4. ยึดเหนี่ยวแท็กเจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 5
5. แปลงข้อมูลฟังก์ชันถ่ายโอนให้เป็นเท็กซ์เจอร์
6. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 1
7. คำนวณเวกเตอร์ปกติของข้อมูลเชิงปริมาตร
8. สร้างเท็กซ์เจอร์ 3 มิติเพื่อเก็บเวกเตอร์ปกติ
9. ยึดเหนี่ยวเท็กซ์เจอร์ของเวกเตอร์ปกติเข้ากับหน่วยเท็กซ์เจอร์ที่ 2
10. สร้างนอร์มอลไลเซชันคิวบ์แม็พ
11. ยึดเหนี่ยวนอร์มอลไลเซชันคิวบ์แม็พเข้ากับหน่วยเท็กซ์เจอร์ที่ 3
12. แปลงเวกเตอร์ทิศทางของแสงเป็นค่าสีแล้วเก็บไว้ในเรจิสเตอร์คงตัว 0 (con0)
13. เก็บค่าความเข้มของแสงแวดล้อมลงในเรจิสเตอร์สีรอง (secondary color)
14. โปรแกรมแฟรกเม็นต์เชดดิงตามรหัสดังรูปที่ 4.6
15. แม็พหน่วยเท็กซ์เจอร์ที่ 0 1 2 3 และ 5 เข้ากับรีแซมพลิงสไลซ์

รหัสโปรแกรมแฟรกเม็นต์เชดดิงในรูปที่ 4.6 จะเริ่มจาก

<u>บรรทัดที่ 3</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 0 ซึ่งเก็บค่าสเกลาร์มาเก็บไว้ในเรจิสเตอร์ 0

<u>บรรทัดที่ 4</u> อ่านต่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 2 ซึ่งเก็บเวกเตอร์ปกติของข้อมูลเชิงปริมาตรมาเก็บไว้ในเรจิสเตอร์ 2

<u>บรรทัดที่ 5</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 5 ซึ่งเก็บค่าฉลากมาเก็บไว้ในเรจิสเตอร์ 5

<u>บรรทัดที่ 7-9</u> ย้ายค่าความทึบแสงของเรจิสเตอร์ 0 มาเก็บที่ในช่องสีแดงของเรจิสเตอร์ 5 ส่งผลให้ในขณะนี้ช่องสีแดงของเรจิสเตอร์ 5 มีค่าสเกลาร์ และช่องสีที่เหลือเป็นค่าฉลาก

<u>บรรทัดที่ 11-13</u> แปลงค่าสีของเรจิสเตอร์ 2 ให้กลายเป็นเวกเตอร์เพื่อเตรียมใช้อ่านค่าจากนอร์มอลไลเซชันคิวบ์แม็พ

<u>จบรอบการทำงานแรก</u>

```
1   glBeginFragmentShaderATI();
2
3     glSampleMapATI(GL_REG_0_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Pri. Volume Data
4     glSampleMapATI(GL_REG_2_ATI, GL_TEXTURE2_ARB, GL_SWIZZLE_STR_ATI);// Sample Normal Vector
5     glSampleMapATI(GL_REG_5_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Sec. Volume Data
6
7     glColorFragmentOp1ATI(GL_MOV_ATI,
8               GL_REG_5_ATI, GL_RED_BIT_ATI, GL_NONE,
9               GL_REG_0_ATI, GL_ALPHA, GL_NONE);
10
11    glColorFragmentOp1ATI(GL_MOV_ATI,
12              GL_REG_2_ATI, GL_NONE, GL_NONE,
13              GL_REG_2_ATI, GL_NONE, GL_BIAS_BIT_ATI|GL_2X_BIT_ATI|GL_NEGATE_BIT_ATI);
14
15    //End of first pass
16    glSampleMapATI(GL_REG_1_ATI, GL_REG_5_ATI,GL_SWIZZLE_STR_ATI); //Sample TF
17    glSampleMapATI(GL_REG_3_ATI, GL_REG_2_ATI,GL_SWIZZLE_STR_ATI); //Sample Cube Normalize
18
19
20    glColorFragmentOp2ATI(GL_DOT3_ATI,
21              GL_REG_0_ATI, GL_NONE, GL_SATURATE_BIT_ATI,
22              GL_REG_3_ATI, GL_NONE, GL_BIAS_BIT_ATI|GL_2X_BIT_ATI,
23              GL_CON_0_ATI, GL_NONE, GL_BIAS_BIT_ATI|GL_2X_BIT_ATI);
24
25    glColorFragmentOp2ATI(GL_ADD_ATI,
26              GL_REG_0_ATI, GL_NONE, GL_NONE,
27              GL_REG_0_ATI, GL_NONE, GL_NONE,
28              GL_SECONDARY_INTERPOLATOR_ATI, GL_NONE, GL_NONE);
29
30    glColorFragmentOp2ATI(GL_MUL_ATI,
31              GL_REG_0_ATI, GL_NONE, GL_SATURATE_BIT_ATI,
32              GL_REG_0_ATI, GL_NONE, GL_NONE,
33              GL_REG_1_ATI, GL_NONE, GL_NONE);
34
35    glAlphaFragmentOp1ATI(GL_MOV_ATI,
36              GL_REG_0_ATI, GL_NONE,
37              GL_REG_1_ATI, GL_ALPHA, GL_NONE);
38
39   glEndFragmentShaderATI();
```

รูปที่ 4.6 รหัสแฟรกเม็นต์เชดดิงในการให้แสงเงาแบบแสงแพร่

บรรทัดที่ 16 อ่านค่าหน่วยเท็กซ์เจอร์ 1 ซึ่งเก็บฟังก์ชันถ่ายโอนโดยใช้ค่าในเรจิสเตอร์ 5 ในการกำหนดตำแหน่งพิกัดในการอ่าน ทำให้เกิดการอ่านค่าแบบดีเพ็นแดนต์ ดังนั้นค่าที่ได้ในเรจิสเตอร์ 1 จึงเป็นค่าสีและความทึบแสงของวัตถุ

บรรทัดที่ 17 อ่านค่าหน่วยเท็กซ์เจอร์ 3 ซึ่งเป็นนอร์มอลไลเซชันคิวบ์แม็พโดยใช้เวกเตอร์ปกติที่ถูกเก็บในเรจิสเตอร์ 2 เป็นตัวชี้บอกตำแหน่ง ส่งผลให้ได้เวกเตอร์ปกติซึ่งเป็นเวกเตอร์หนึ่งหน่วยเก็บไว้ในเรจิสเตอร์ 3

บรรทัดที่ 20-23 คูณจุดค่าเวกเตอร์ปกติในเรจิสเตอร์ 3 กับค่าทิศทางของแสงซึ่งเก็บอยู่ในเรจิสเตอร์สีคงตัว 0 ผลที่ได้เก็บไว้ในเรจิสเตอร์ 0

บรรทัดที่ 25-28 รวมผลจากการคูณจุดกับค่าความเข้มของแสงแวดล้อมที่เก็บอยู่ในเรจิสเตอร์สีรอง เก็บผลที่ได้ในเรจิสเตอร์ 0

บรรทัดที่ 30-33 คูณผลการรวมในเรจิสเตอร์ 0 กับค่าสีของวัตถุในเรจิสเตอร์หนึ่ง ค่าที่ได้เป็นค่าสีผลลัพธ์จากสมการที่ 4.1 เก็บไว้ในเรจิสเตอร์ 0 ซึ่งเป็นเรจิสเตอร์ส่งออก

<u>บรรทัดที่ 35-37</u> ย้ายค่าความทึบแสงจากเรจิสเตอร์ 1 มาเก็บไว้ในเรจิสเตอร์ 0 ซึ่งเป็นเรจิสเตอร์ส่งออก

4.4 ขั้นตอนวิธีในการให้แสงเงาแบบแสงกล้า

ขั้นตอนวิธีในการสร้างภาพโดยตรงนี้ถูกใช้ใน ภาวะ 4 ของส่วนโปรแกรม VolumeX โดยข้อมูลนำเข้าจะเหมือนกับในขั้นตอนวิธีการให้แสงเงาแบบแสงแพร่ เพียงเพิ่มทิศทางของผู้สังเกตเข้ามา จากสมการที่ 2.19 ให้สัมประสิทธิ์ของการสะท้อนแสงแวดล้อมและแสงแพร่เท่ากันคือสีของวัตถุ และให้สัมประสิทธิ์ของการสะท้อนแสงกล้าของวัตถุเท่ากับ 1 จะสามารถเขียนสมการใหม่ได้เป็น

$$I = base \times (I_a + N \bullet L) + (N \bullet H)^n \qquad ...(4.5)$$

เมื่อ I คือความเข้มของแสงที่วัตถุสะท้อนออกมา

$base$ คือค่าสีของพื้นผิววัตถุ

I_a คือความเข้มของแสงแวดล้อม

N คือเวกเตอร์ปกติของพื้นผิว

L คือทิศทางของแสง

H คือเวกเตอร์ที่ระหว่างกลางของทิศทางของแสงและทิศทางของผู้สังเกตุ

n คือค่าดัชนีที่ใช้จำลองความเรียบมันของวัตถุ

เวกเตอร์ H สามารถคำนวณได้ตามสมการที่ 2.17 เนื่องจากทั้งทิศทางของแสงและทิศทางของผู้สังเกตเป็นค่าคงตัวซึ่งไม่ขึ้นกับข้อมูลเชิงปริมาตรดังนั้นเวกเตอร์ H จึงสามารถนอร์มอลไลซ์ได้ก่อนที่จะป้อนให้กราฟิกส์ฮาร์ดแวร์

เนื่องจากข้อจำกัดของกราฟิกส์ฮาร์ดแวร์ของเอ็นวิเดียค่าดัชนีที่ใช้จำลองความเรียบมันของวัตถุที่ใช้ในงานวิจัยนี้จะมีค่าคงที่ที่ 128

เช่นเดียวกับการสร้างภาพปกติและการให้แสงเงาแบบแสงแพร่ขั้นตอนวิธีการทำงานสามารถแบ่งเป็น 2 หัวข้อ ได้แก่

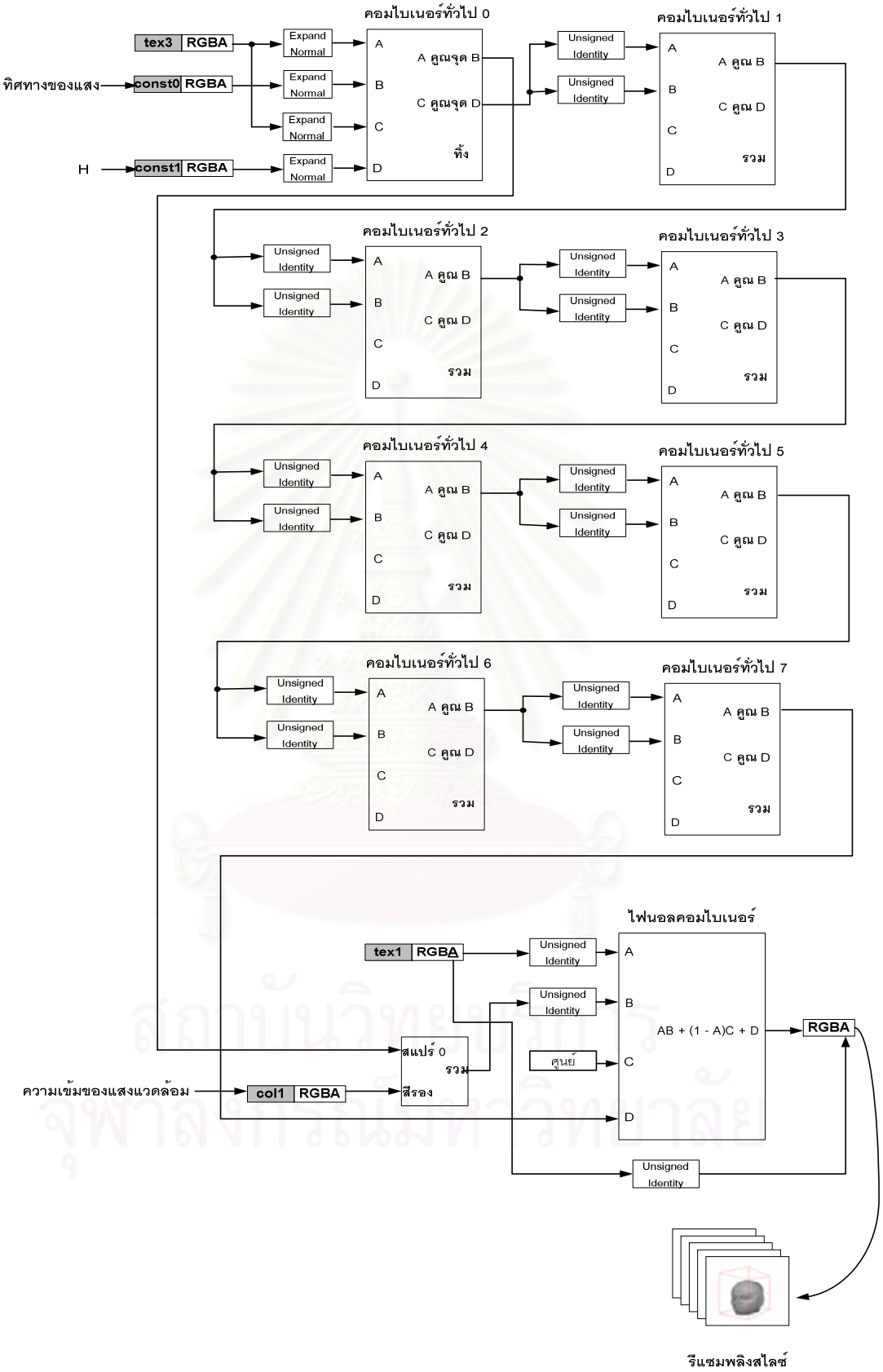
4.4.1 ขั้นตอนวิธีในการให้แสงเงาแบบแสงกล้าบนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์

บนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์ขั้นตอนวิธีในการให้แสงเงาแบบแสงกล้าจะเหมือนกับขั้นตอนวิธีในการให้แสงเงาแบบแสงแพร่ โดยเฉพาะในขั้นตอนของเท็กซ์เจอร์เชดเดอร์ จะต่างกันเพียงในส่วนของเรจิสเตอร์คอมไบเนอร์ และเพิ่มการส่งค่าเวกเตอร์ H ไปยังกราฟิกส์ฮาร์ดแวร์ ซึ่งจะมีขั้นตอนวิธีดังต่อไปนี้

1. แปลงข้อมูลค่าสเกลาร์และข้อมูลค่าฉลากให้เป็นเท็กซ์เจอร์ 3 มิติ โดยให้ค่าสเกลาร์อยู่ในช่องความทึบแสง (A) และข้อมูลค่าฉลากอยู่ในช่องสีแดง (R)

2. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้เข้ากับหน่วยเท็กซ์เจอร์ที่ 0
3. แปลงข้อมูลฟังก์ชันถ่ายโอนให้เป็นเท็กซ์เจอร์
4. ยึดเหนี่ยวเท็กซ์เจอร์ของฟังก์ชันถ่ายโอนเข้ากับหน่วยเท็กซ์เจอร์ที่ 1
5. คำนวณเวกเตอร์ปกติของข้อมูลเชิงปริมาตร
6. สร้างเท็กซ์เจอร์ 3 มิติเพื่อเก็บเวกเตอร์ปกติ
7. ยึดเหนี่ยวเท็กซ์เจอร์ของเวกเตอร์ปกติเข้ากับหน่วยเท็กซ์เจอร์ที่ 2
8. สร้างนอร์มอลไลเซชันคิวบ์แม็พ
9. ยึดเหนี่ยวนอร์มอลไลเซชันคิวบ์แม็พเข้ากับหน่วยเท็กซ์เจอร์ที่ 3
10. แปลงเวกเตอร์ทิศทางของแสงเป็นค่าสีแล้วเก็บไว้ในเรจิสเตอร์คงตัว 0 (const0)
11. คำนวณหาเวกเตอร์ H แล้วแปลงเป็นค่าสีจากนั้นเก็บไว้ในเรจิสเตอร์คงตัว 1 (const1)
12. เก็บค่าความเข้มของแสงแวดล้อมลงในเรจิสเตอร์สีรอง (col1)
13. กำหนดให้เท็กซ์เจอร์เซดเดอร์ของหน่วยเท็กซ์เจอร์ที่ 1 ใช้ตัวดำเนินการ Dependent Alpha-Red Texturing ในการอ่านค่า
14. กำหนดให้เท็กซ์เจอร์เซดเดอร์ของหน่วยเท็กซ์เจอร์ที่ 3 ใช้ตัวดำเนินการ Dependent RGB Texuture Cube Map
15. โปรแกรมเรจิสเตอร์คอมไบเนอร์ให้คำนวณตามสมการที่ 4.5 ดังรูปที่ 4.7
16. แม็พหน่วยเท็กซ์เจอร์ที่ 0 1 2 และ 3 เข้ากับรีแซมพลิงสไลซ์

ในส่วนเท็กซ์เจอร์เซดเดอร์นั้นการให้แสงเงาแบบแสงกล้าการอ่านเท็กซ์เจอร์ยังคงเหมือนในรูปที่ 4.5 ก แต่ในส่วนเรจิสเตอร์คอมไบเนอร์นั้นจะเปลี่ยนมาใช้ตามรูปที่ 4.7 โดยคอมไบเนอร์ทั่วไปที่ 0 จะคำนวณการคูณจุดของทิศทางของแสงกับเวกเตอร์ปกติ และคูณจุดระหว่างเวกเตอร์ H กับเวกเตอร์ปกติ โดยผลการคูณจุดระหว่างเวกเตอร์ H และเวกเตอร์ปกติจะถูกนำไปยกกำลังโดยใช้การคูณในคอมไบเนอร์ทั่วไปที่ 1 ถึง 7 ผลของการคูณจะได้เท่ากับผลการคูณจุดระหว่างเวกเตอร์ H และเวกเตอร์ปกติยกกำลัง 128 ค่านี้จะถูกส่งเข้าไฟนอลคอมไบเนอร์ทางช่อง D เพื่อรอรวมกับผลคูณของค่าสีของวัตถุที่ช่อง A กับผลรวมของการคูณจุดระหว่างเวกเตอร์ปกติกับทิศทางของแสงกับความเข้มของแสงแวดล้อมที่ช่อง B เมื่อคำนวณเสร็จแล้วค่าสีที่ได้จะเป็นไปตามการคำนวณของสมการที่ 4.5 โดยค่าความทึบแสงจะถูกส่งผ่านโดยตรงโดยไม่มีการคำนวณ



รูปที่ 4.7 ส่วนเรจิสเตอร์คอมไบเนอร์ในการให้แสงเงาแบบแสงกล้าของเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์

4.4.2 ขั้นตอนวิธีในการให้แสงเงาแบบแสงกล้าบนเอทีไอกราฟิกส์ฮาร์ดแวร์

บนเอทีไอกราฟิกส์ฮาร์ดแวร์การหาค่าพจน์ของแสงกล้า $(N \bullet H)^n$ จะแตกต่างจากบนเอ็นวิเดียกราฟิกส์ฮาร์ดแวร์ โดยจะใช้การสร้างตารางการค้นหาค่าจากฟังก์ชันการคำนวณที่สร้างขึ้นในการหาค่าพจน์ของแสงกล้าในกรณีที่เวกเตอร์ปกติไม่เป็นเวกเตอร์หนึ่งหน่วย จะได้

$$f(x,y) = \left(\frac{N \bullet H}{|N|} \right)^n \qquad \text{...(4.6)}$$

แทนค่า $|N|$ ด้วย

$$|N| = \sqrt{|N|^2} = \sqrt{|N||N|\cos 0} = \sqrt{N \bullet N} \qquad \text{...(4.7)}$$

จะได้

$$f(x,y) = \left(\frac{N \bullet H}{\sqrt{N \bullet N}} \right)^n \qquad \text{...(4.8)}$$

ดังนั้นจึงสามารถเขียน $f(x,y)$ ให้อยู่ในรูปของ $f(N \bullet H, N \bullet N)$ ได้ดังสมการที่ 4.8 จากนั้นกำหนดให้โดเมนของ $N \bullet H$ อยู่ในช่วง 0 ถึง 1 และโดเมนของ $N \bullet N$ อยู่ในช่วง 0 ถึง 1 จะสามารถสร้างตารางค้นหาค่าของ $f(N \bullet H, N \bullet N)$ นี้ได้โดยใช้สมการที่ 4.8 เมื่อกำหนดค่า n เป็นค่าคงตัวค่าหนึ่ง ในงานวิจัยนี้จะกำหนดให้ขนาดของตารางค้นหาค่าเท่ากับ 1024×1024 และค่า n เท่ากับ 128 จากนั้นจึงแปลงตารางค้นหาค่านี้ให้เป็นเท็กซ์เจอร์เพื่อให้สามารถค้นค่าโดยใช้กราฟิกส์ฮาร์ดแวร์ได้

ขั้นตอนวิธีการสร้างภาพบนเอทีไอกราฟิกส์ฮาร์ดแวร์ มีดังนี้

1. แปลงข้อมูลค่าสเกลาร์ให้เป็นเท็กซ์เจอร์ 3 มิติ
2. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 0
3. แปลงข้อมูลค่าฉลากให้เป็นเท็กซ์เจอร์ 3 มิติ
4. ยึดเหนี่ยวแท็กเจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 5
5. แปลงข้อมูลฟังก์ชันถ่ายโอนให้เป็นเท็กซ์เจอร์
6. ยึดเหนี่ยวเท็กซ์เจอร์ที่ได้กับหน่วยเท็กซ์เจอร์ที่ 1
7. คำนวณเวกเตอร์ปกติของข้อมูลเชิงปริมาตร
8. สร้างเท็กซ์เจอร์ 3 มิติเพื่อเก็บเวกเตอร์ปกติ
9. ยึดเหนี่ยวเท็กซ์เจอร์ของเวกเตอร์ปกติเข้ากับหน่วยเท็กซ์เจอร์ที่ 2
10. สร้างนอร์มอลไลเซชันคิวบ์แม็พ
11. ยึดเหนี่ยวนอร์มอลไลเซชันคิวบ์แม็พเข้ากับหน่วยเท็กซ์เจอร์ที่ 3
12. สร้างตารางการค้นหาค่าพจน์ของแสงกล้า

13. แปลงตารางการค้นหาค่าให้เป็นเท็กซ์เจอร์แล้วยึดเหนี่ยวเข้ากับหน่วยเท็กซ์เจอร์ที่ 4
14. แปลงเวกเตอร์ทิศทางของแสงเป็นค่าสีแล้วเก็บไว้ในเรจิสเตอร์คงตัว 0 (con0)
15. คำนวณค่าเวกตอร์ H แปลงเป็นค่าสีแล้วเก็บไว้ในเรจิสเตอร์คงตัว 1 (con1)
16. เก็บค่าความเข้มของแสงแวดล้อมลงในเรจิสเตอร์สีรอง (secondary color)
17. โปรแกรมแฟรกเม็นต์เชดดิงตามรหัสดังรูปที่ 4.6
18. แม็พหน่วยเท็กซ์เจอร์ที่ 0 1 2 3 4 และ 5 เข้ากับรีแซมพลิงสไลซ์

รหัสโปรแกรมแฟรกเม็นต์เชดดิงในรูปที่ 4.8 จะเริ่มจาก

<u>บรรทัดที่ 3</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 0 ซึ่งเก็บค่าสเกลาร์มาเก็บไว้ในเรจิสเตอร์ 0

<u>บรรทัดที่ 4</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 2 ซึ่งเก็บเวกเตอร์ปกติของข้อมูลเชิงปริมาตรมาเก็บไว้ในเรจิสเตอร์ 2

<u>บรรทัดที่ 5</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 5 ซึ่งเก็บค่าฉลากมาเก็บไว้ในเรจิสเตอร์ 5

<u>บรรทัดที่ 8-10</u> ย้ายค่าความทึบแสงของเรจิสเตอร์ 0 มาเก็บที่ในช่องสีแดงของเรจิสเตอร์ 5 ส่งผลให้ในขณะนี้ช่องสีแดงของเรจิสเตอร์ 5 มีค่าสเกลาร์ และช่องสีที่เหลือเป็นค่าฉลาก

<u>บรรทัดที่ 12-14</u> แปลงค่าสีของเรจิสเตอร์ 2 ให้กลายเป็นเวกเตอร์เพื่อเตรียมใช้อ่านค่าจากนอร์มอลไลเซชันคิวบ์แม็พ

<u>บรรทัดที่ 17-20</u> หาค่าคูณจุดของเวกเตอร์ปกติกับเวกเตอร์กึ่งกลางระหว่างทิศทางของแสงกับทิศทางของผู้สังเกตุ $(N \bullet H)$ เก็บค่าที่ได้ไว้ในเรจิสเตอร์ 1 ช่องสีแดงเพื่อเตรียมใช้อ่านค่าจากตารางค้นหาค่า

<u>บรรทัดที่ 23-26</u> หาค่าคูณจุดของเวกเตอร์ปกติกับเวกเตอร์ปกติ $(N \bullet N)$ เก็บค่าที่ได้ไว้ในเรจิสเตอร์ 1 ช่องสีเขียวเพื่อเตรียมใช้อ่านค่าจากตารางค้นหาค่า

<u>จบรอบการทำงานแรก</u>

<u>บรรดทัดที่ 29</u> อ่านค่าเท็กซ์เจอร์ของหน่วยเท็กซ์เจอร์ที่ 4 ซึ่งเก็บตารางค้นหาค่า โดยใช้ค่า $(N \bullet H)$ และ $(N \bullet N)$ ในเรจิสเตอร์ 1 เป็นตัวระบุตำแหน่งในการอ่านค่า ผลที่ได้จะเป็นค่าประมาณของฟังก์ชัน $f(x, y)$ ในสมการที่ 4.6

<u>บรรทัดที่ 30</u> อ่านค่าหน่วยเท็กซ์เจอร์ 1 ซึ่งเก็บฟังก์ชันถ่ายโอนโดยใช้ค่าในเรจิสเตอร์ 5 ในการกำหนดตำแหน่งพิกัดในการอ่าน ทำให้เกิดการอ่านค่าแบบดีเพ็นแดนต์ ดังนั้นค่าที่ได้ในเรจิสเตอร์ 1 จึงเป็นค่าสีและความทึบแสงของวัตถุ

<u>บรรทัดที่ 31</u> อ่านค่าหน่วยเท็กซ์เจอร์ 3 ซึ่งเป็นนอร์มอลไลเซชันคิวบ์แม็พโดยใช้เวกเตอร์ปกติที่ถูกเก็บในเรจิสเตอร์ 2 เป็นตัวชี้บอกตำแหน่ง ส่งผลให้ได้เวกเตอร์ปกติ

ซึ่งเป็นเวกเตอร์หนึ่งหน่วยเก็บไว้ในเรจิสเตอร์ 3

บรรทัดที่ 34-37 คูณจุดค่าเวกเตอร์ปกติในเรจิสเตอร์ 3 กับค่าทิศทางของแสงซึ่งเก็บอยู่ในเรจิสเตอร์สีคงตัว 0 ผลที่ได้เก็บไว้ในเรจิสเตอร์ 0

บรรทัดที่ 40-43 รวมผลจากการคูณจุดกับค่าความเข้มของแสงแวดล้อมที่เก็บอยู่ในเรจิสเตอร์สีรอง เก็บผลที่ได้ในเรจิสเตอร์ 0

บรรทัดที่ 46-50 คูณผลการรวมในเรจิสเตอร์ 0 กับค่าสีของวัตถุในเรจิสเตอร์หนึ่ง แล้วรวมกับค่าที่ได้จากตารางค้นหาค่าในเรจิสเตอร์ 4 ค่าที่ได้เป็นค่าสีผลลัพธ์จากสมการที่ 4.5 เก็บไว้ในเรจิสเตอร์ 0 ซึ่งเป็นเรจิสเตอร์ส่งออก

บรรทัดที่ 52-54 ย้ายค่าความทึบแสงจากเรจิสเตอร์ 1 มาเก็บไว้ในเรจิสเตอร์ 0 ซึ่งเป็นเรจิสเตอร์ส่งออก

```
1   glBeginFragmentShaderATI();
2
3           glSampleMapATI(GL_REG_0_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Pri. Volume Data
4           glSampleMapATI(GL_REG_2_ATI, GL_TEXTURE2_ARB, GL_SWIZZLE_STR_ATI);// Sample Normal Vector
5           glSampleMapATI(GL_REG_5_ATI, GL_TEXTURE0_ARB, GL_SWIZZLE_STR_ATI);// Sample Sec.
6
7
8           glColorFragmentOp1ATI(GL_MOV_ATI,
9                                 GL_REG_5_ATI, GL_RED_BIT_ATI, GL_NONE,
10                                GL_REG_0_ATI, GL_ALPHA, GL_NONE);
11
12          glColorFragmentOp1ATI(GL_MOV_ATI,
13                                GL_REG_2_ATI, GL_NONE, GL_NONE,
14                                GL_REG_2_ATI, GL_NONE, GL_BIAS_BIT_ATI|GL_2X_BIT_ATI|GL_NEGATE_BIT_ATI);
15
16          //N.H
17          glColorFragmentOp2ATI(GL_DOT3_ATI,
18                                GL_REG_1_ATI, GL_RED_BIT_ATI, GL_SATURATE_BIT_ATI,
19                                GL_REG_2_ATI, GL_NONE, GL_NONE,
20                                GL_CON_1_ATI, GL_NONE, GL_BIAS_BIT_ATI|GL_2X_BIT_ATI);
21
22          //N.N
23          glColorFragmentOp2ATI(GL_DOT3_ATI,
24                                GL_REG_1_ATI, GL_GREEN_BIT_ATI, GL_SATURATE_BIT_ATI,
25                                GL_REG_2_ATI, GL_NONE, GL_NONE,
26                                GL_REG_2_ATI, GL_NONE, GL_NONE);
27
28          // End of first pass
29          glSampleMapATI(GL_REG_4_ATI, GL_REG_1_ATI,GL_SWIZZLE_STR_ATI); //Sample NHNN (N.H)^k
30          glSampleMapATI(GL_REG_1_ATI, GL_REG_5_ATI,GL_SWIZZLE_STR_ATI); //Sample TF (Base)
31          glSampleMapATI(GL_REG_3_ATI, GL_REG_2_ATI,GL_SWIZZLE_STR_ATI); //Sample Cube Normalize (N)
32
33          // N.L
34          glColorFragmentOp2ATI(GL_DOT3_ATI,
35                                GL_REG_0_ATI, GL_NONE, GL_SATURATE_BIT_ATI,
36                                GL_REG_3_ATI, GL_NONE, GL_BIAS_BIT_ATI|GL_2X_BIT_ATI,
37                                GL_CON_0_ATI, GL_NONE, GL_BIAS_BIT_ATI|GL_2X_BIT_ATI);
38
39          // N.L + Ambient
40          glColorFragmentOp2ATI(GL_ADD_ATI,
41                                GL_REG_0_ATI, GL_NONE, GL_NONE,
42                                GL_REG_0_ATI, GL_NONE, GL_NONE,
43                                GL_SECONDARY_INTERPOLATOR_ATI, GL_NONE, GL_NONE);
44
45          // (N.L + Ambient) * Base) + (N.H)^k
46          glColorFragmentOp3ATI(GL_MAD_ATI,
47                                GL_REG_0_ATI, GL_NONE, GL_SATURATE_BIT_ATI,
48                                GL_REG_0_ATI, GL_NONE, GL_NONE,
49                                GL_REG_1_ATI, GL_NONE, GL_NONE,
50                                GL_REG_4_ATI, GL_NONE, GL_NONE);
51
52          glAlphaFragmentOp1ATI(GL_MOV_ATI,
53                                GL_REG_0_ATI, GL_NONE,
54                                GL_REG_1_ATI, GL_ALPHA, GL_NONE);
55
56
57  glEndFragmentShaderATI();
```

รูปที่ 4.8 รหัสแฟรกเม็นต์เชดดิงในการให้แสงเงาแบบแสงกล้า

4.5 ขั้นตอนวิธีในการตัด

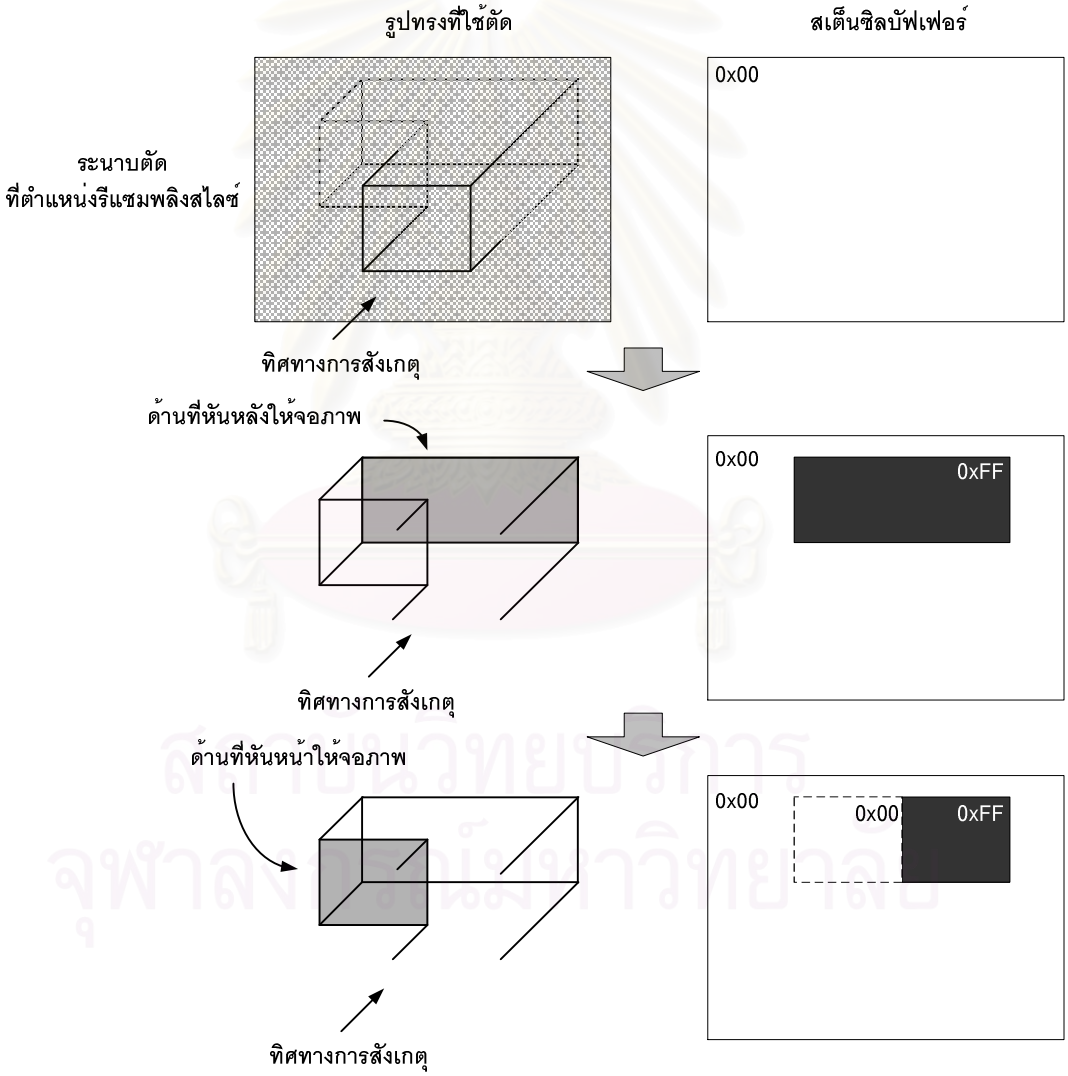
ขั้นตอนวิธีในการตัดที่กล่าวถึงในหัวข้อนี้เป็นขั้นตอนวิธีเฉพาะการตัดโดยใช้รูปทรงเนื่องจากการตัดโดยใช้ระนาบนั้นเป็นความสามารถมาตรฐานของโอเพ็นจีแอล หลักการของการตัดโดยใช้รูปทรงนั้นคือก่อนการวาดรีแซมพลิงสไลซ์แต่ละแผ่นต้องแยกส่วนของรีแซมพลิงสไลซ์ที่อยู่ภายในและภายนอกรูปทรงที่ต้องการตัดออกจากกันให้ได้แล้ววาดภาพลงบนรีแซมพลิงสไลซ์เฉพาะส่วนที่ต้องการ

ในการแยกว่าตำแหน่งบนรีแซมพลิงสไลซ์อยู่ภายในหรือภายนอกรูปทรงที่ใช้ตัดทำได้โดยใช้ความสามารถในการทดสอบคัลแฟซ และการทดสอบสเต็นซิลของกราฟิกส์ฮาร์ดแวร์ (2.2.2)

ขั้นตอนวิธีในการตัดโดยใช้รูปทรงมี ดังนี้

1. วางระนาบตัด ณ ตำแหน่งที่ต้องการวาดรีแซมพลิงสไลซ์เพื่อตัดรูปทรงที่ใช้ตัดด้านที่อยู่ใกล้จอภาพออก
2. ตั้งการทดสอบคัลแฟซให้ตัดรูปหลายเหลี่ยมที่หันหน้าเข้าหาจอภาพทิ้ง
3. ตั้งสเต็นซิลฟังก์ชันให้ ผ่านตลอด ค่าอ้างอิงเท่ากับ 0xFF และค่ามาสก์เท่ากับ 0xFF
4. ตั้งการดำเนินการสเต็นซิลให้ทุกกรณีเป็นการคงค่าเดิม ยกเว้นกรณีที่แฟรกเม็นต์ผ่านทั้งการทดสอบสเต็นซิล และการทดสอบความลึก ให้แทนที่ค่าในสเต็นซิลบัฟเฟอร์ด้วยค่าอ้างอิง
5. ปิดการเขียนข้อมูลลงเฟรมบัฟเฟอร์
6. วาดรูปทรงที่ใช้ตัด ซึ่งทำให้ข้อมูลบนสเต็นซิลบัฟเฟอร์ในตำแหน่งที่เห็นด้านหลังของรูปทรงหลายเหลี่ยมจะถูกแทนที่ด้วยค่าอ้างอิง (0xFF)
7. ตั้งการทดสอบคัลแฟซให้ตัดรูปหลายเหลี่ยมที่หันหลังเข้าหาจอภาพทิ้ง
8. ตั้งสเต็นซิลฟังก์ชันให้ ผ่านตลอด ค่าอ้างอิงเท่ากับ 0x00 และค่ามาสก์เท่ากับ 0xFF
9. ตั้งการดำเนินการสเต็นซิลให้ทุกกรณีเป็นการคงค่าเดิม ยกเว้นกรณีที่แฟรกเม็นต์ผ่านทั้งการทดสอบสเต็นซิล และการทดสอบความลึก ให้แทนที่ค่าในสเต็นซิลบัฟเฟอร์ด้วยศูนย์ (0x00)
10. วาดรูปทรงที่ใช้ตัด ซึ่งทำให้ข้อมูลบนสเต็นซิลบัฟเฟอร์ในตำแหน่งที่เห็นด้านหน้าของรูปทรงหลายเหลี่ยมจะถูกแทนที่ด้วยค่าศูนย์ (0x00)
11. ตั้งสเต็นซิลฟังก์ชันให้เป็น เท่ากับ ในกรณีที่ผู้ใช้ต้องการวาดภาพเฉพาะส่วนที่อยู่นอกรูปทรงที่ใช้ตัด และไม่เท่ากับ ในกรณีที่ผู้ใช้ต้องการวาดภาพเฉพาะส่วนที่อยู่ภายในรูปทรงที่ใช้ตัด ค่าอ้างอิงเท่ากับ 0x00 และค่ามาสก์เท่ากับ 0xFF
12. ตั้งการดำเนินการสเต็นซิลให้ทุกกรณีเป็นการคงค่าเดิม
13. เปิดการเขียนข้อมูลลงเฟรมบัฟเฟอร์
14. วาดรีแซมพลิงสไลซ์
15. กลับไปเริ่มทำข้อ 1 กับทุกรีแซมพลิงสไลซ์

ภาพรวมของการทำงานแสดงในรูปที่ 4.9 ด้านซ้ายแสดงเป็นตำแหน่งและรูปร่างของรูปทรงที่ใช้ตัด ด้านขวาแสดงข้อมูลในสเต็นซิลบัฟเฟอร์ ขั้นตอนการทำงานเริ่มจากด้านบนเมื่อข้อมูลในสเต็นซิลบัฟเฟอร์ยังเป็นค่าศูนย์ทั้งหมด และรูปทรงที่ใช้ตัดถูกระนาบตัดที่ตำแหน่งวาดรีแซมพลิงสไลซ์ตัดด้านที่อยู่ใกล้ผู้สังเกตออก ถัดไปเป็นการวาดรูปทรงที่ใช้ตัดด้านที่หันหลังให้จอภาพ ค่าบนสเต็นซิลบัฟเฟอร์ที่ตำแหน่งเดียวกับด้านที่หันหลังให้จอภาพจะกลายเป็นค่าอ้างอิง (0xFF) ถัดไปเป็นการวาดรูปทรงที่ใช้ตัดด้านที่หันหน้าให้จอภาพ ค่าบนสเต็นซิลบัฟเฟอร์ที่ตำแหน่งเดียวกับด้านที่หันหน้าให้จอภาพจะกลายเป็นค่าศูนย์ เมื่อเสร็จขั้นตอนนี้แล้วจะได้ว่าที่รีแซมพลิงสไลซ์นั้นตำแหน่งที่ค่าบนสเต็นซิลบัฟเฟอร์เป็นศูนย์จะอยู่ภายนอกรูปทรงที่ใช้ตัดและตำแหน่งที่มีค่าเป็นค่าอ้างอิง (0xFF) จะอยู่ภายในรูปทรงที่ใช้ตัด จากนั้นจึงเลือกวาดภาพตามที่ผู้ใช้ต้องการ



รูปที่ 4.9 ขั้นตอนวิธีการตัดโดยใช้รูปทรง

บทที่ 5

การทดลองและผลการทดลอง

การทดลองในงานวิจัยแบ่งงานออกได้เป็น 3 ส่วนคือ การทดลองสร้างโปรแกรมประยุกต์โดยใช้ส่วนโปรแกรมที่สร้างขึ้น ทดลองความสามารถในการสร้างภาพในแบบต่าง ๆ รวมถึงการทดลองการตัด การทดลองวัดประสิทธิภาพในการสร้างภาพของส่วนโปรแกรม ซึ่งมีรายละเอียดดังนี้

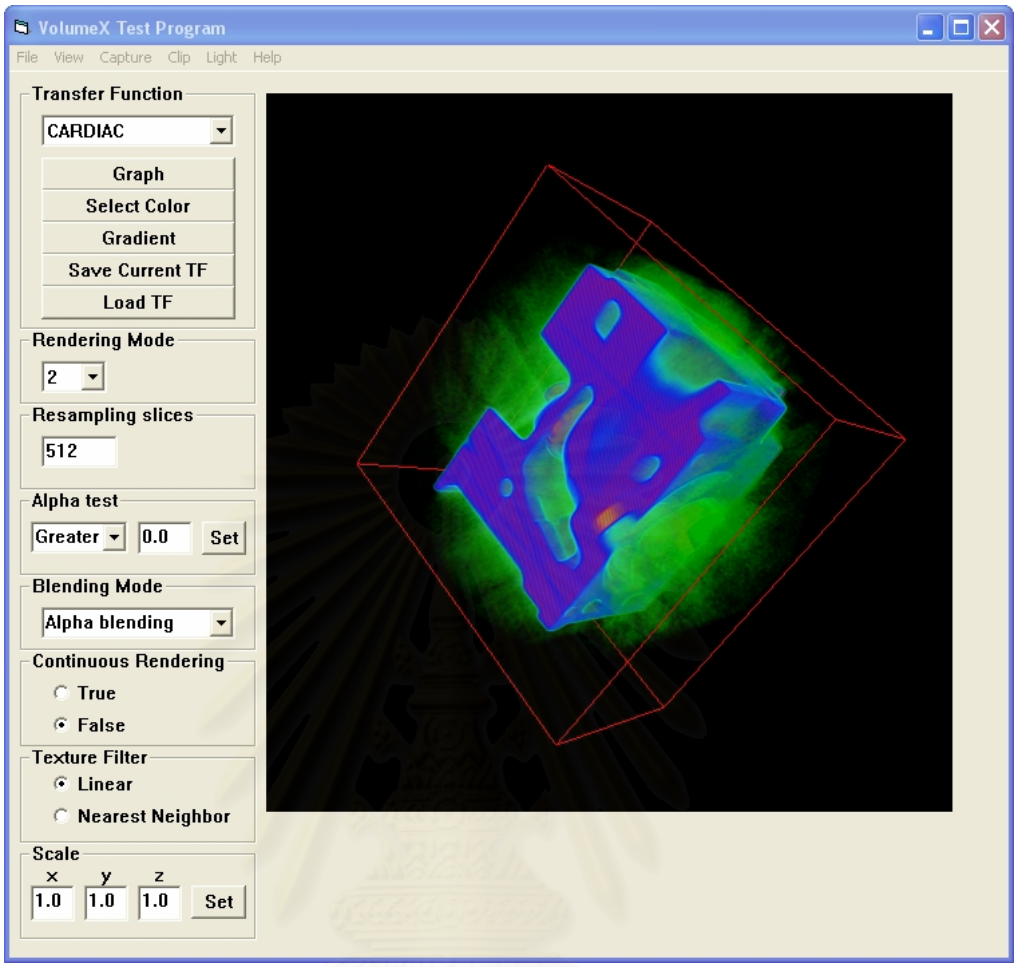
5.1 การทดลองสร้างโปรแกรมประยุกต์โดยใช้ส่วนโปรแกรม

งานวิจัยนี้สร้างโปรแกรมประยุกต์ขึ้น 2 โปรแกรมเพื่อใช้ทดสอบการนำชุดส่วนโปรแกรมไปใช้งาน ได้แก่

5.1.1 โปรแกรมประยุกต์ VolumeX Test Program

โปรแกรมประยุกต์ VolumeX Test Program พัฒนาขึ้นโดยใช้เครื่องมือ Microsoft Visual Basic 5 โปรแกรมประยุกต์นี้สามารถใช้ได้บนเครื่องคอมพิวเตอร์ส่วนบุคคลที่ใช้ระบบปฏิบัติการไมโครซอฟต์วินโดว์ส่วนต่อประสานกับผู้ใช้หลักของโปรแกรมประยุกต์นี้แสดงดังรูปที่ 5.1 คุณสมบัติของโปรแกรมประยุกต์นี้ ได้แก่

1. สามารถเปิดข้อมูลเชิงปริมาตรจากแฟ้มข้อมูลดิบได้
2. สามารถรับข้อมูลได้ทั้งแบบมีเฉพาะค่าสเกลาร์เพียงอย่างเดียว หรือแบบที่มีค่าฉลากด้วย
3. มีฟังก์ชันถ่ายโอนมาตรฐานให้ใช้จำนวน 14 ฟังก์ชัน อนุญาตให้ผู้ใช้สร้างฟังก์ชันถ่ายโอนได้ทั้งในแบบกราฟ เลือกสี หรือเลือกโทนสี สามารถบันทึกและเรียกฟังก์ชันถ่ายโอนจากแฟ้มข้อมูลได้
4. สามารถสร้างภาพได้ทั้งในแบบปกติและแบบมีการให้แสงเงา
5. สนับสนุนการตัดโดยใช้ระนาบและรูปทรง
6. สามารถจับภาพลงในคลิปบอร์ดและแฟ้ม BMP

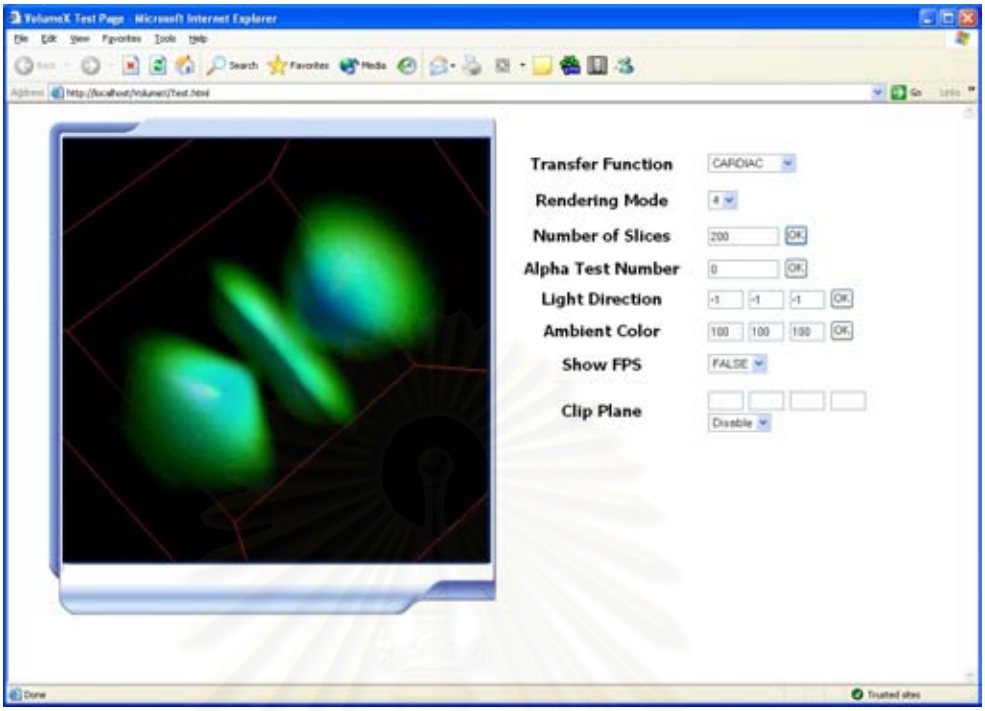


รูปที่ 5.1 ส่วนต่อประสานกับผู้ใช้หลักของโปรแกรมประยุกต์ VolumeX Test Program

5.1.2 เว็บเพจ VolumeX Test Page

เวปเพจ VolumeX Test Page เป็นเว็บเพจที่สร้างขึ้นเพื่อทดสอบการทำงานของชุดส่วนโปรแกรมบนอินเทอร์เน็ตและใช้เว็บเบราว์เซอร์เป็นคอนโทรลคอนเทนเนอร์ โดยเครื่องมือที่ใช้การพัฒนาคือ ASP และ VBScript ตัวบริการเว็บใช้ Microsoft Internet Information Services 5 ร่วมกับเว็บเบราว์เซอร์ Microsoft Internet Explorer 6 หน้าเว็บเพจนี้แสดงในรูปที่ 5.2 คุณสมบัติของเว็บเพจนี้ คือ

1. สามารถสร้างภาพเชิงปริมาตรทั้งแบบในแบบปกติและให้แสงเงาได้
2. มีฟังก์ชันถ่ายโอนมาตรฐานให้เลือกใช้ 14 ฟังก์ชัน
3. สามารถใช้การตัดแบบระนาบได้



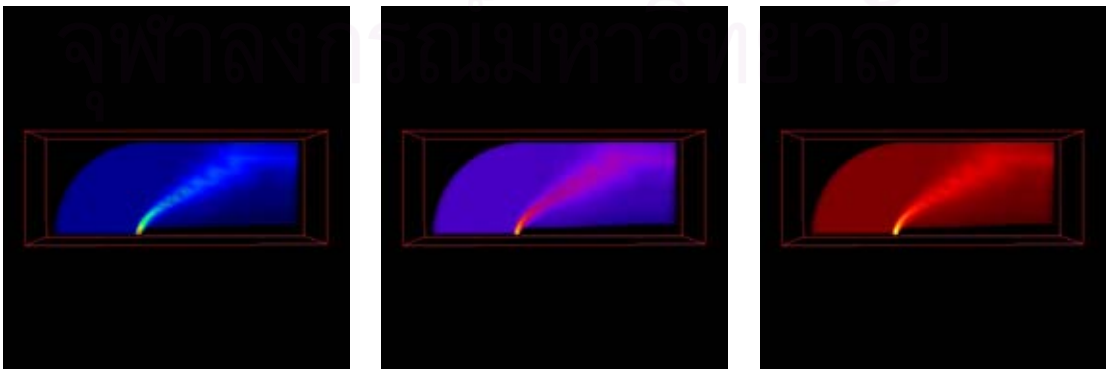
รูปที่ 5.2 เว็บเพจ VolumeX Test Page

5.2 การทดลองการสร้างภาพ

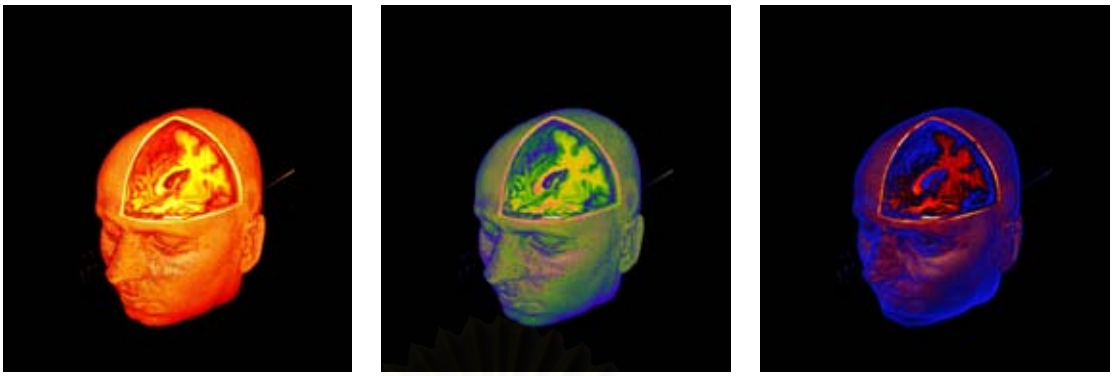
ในหัวข้อนี้จะแสดงให้เห็นถึงภาพที่ได้จากการสร้างภาพเชิงปริมาตรโดยใช้ความสามารถของชุดส่วนโปรแกรมที่งานวิจัยนี้ได้สร้างขึ้นในรูปแบบต่าง ๆ คือ การเปลี่ยนฟังก์ชันถ่ายโอน การเปลี่ยนการให้แสงเงา การใช้ข้อมูลนำเข้าแบบที่มีทั้งค่าสเกลาร์และค่าฉลาก การตัดโดยใช้ระนาบและรูปทรง

5.2.1 การเปลี่ยนฟังก์ชันถ่ายโอน

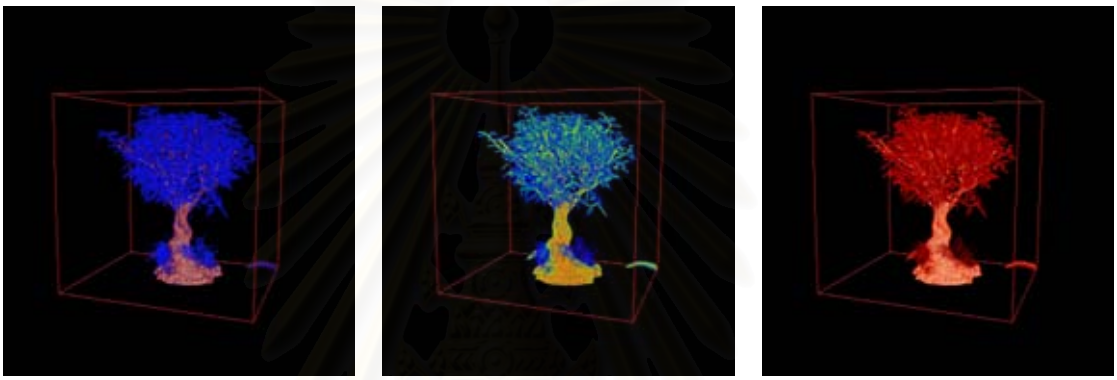
การทดลองเปลี่ยนฟังก์ชันถ่ายโอนโดยได้ทดสอบกับชุดข้อมูลเชิงปริมาตรจำนวน 3 ชุด แต่ละชุดทดลองเปลี่ยนฟังก์ชันถ่ายโอนจำนวน 3 ฟังก์ชันเพื่อดูความเปลี่ยนแปลงที่เกิดขึ้นกับภาพที่สร้างได้ ดังรูปที่ 5.3 5.4 และ 5.5 สำหรับเวลาที่ใช้ในการเปลี่ยนฟังก์ชันถ่ายโอนนั้นน้อยมากจนสังเกตไม่ได้



รูปที่ 5.3 แบบจำลองการไหลของอากาศเมื่อใช้ฟังก์ชันถ่ายโอนที่แตกต่างกัน



รูปที่ 5.4 ภาพ MR ของศีรษะมนุษย์เมื่อใช้ฟังก์ชันถ่ายโอนแตกต่างกัน



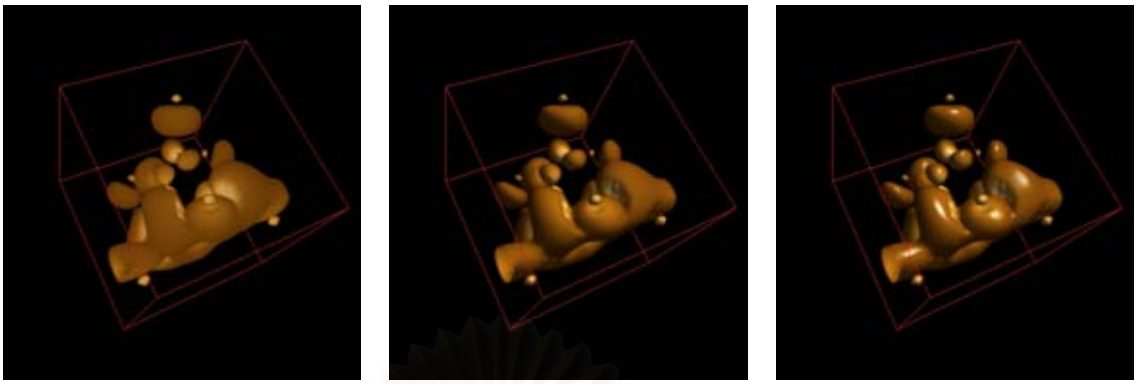
รูปที่ 5.5 ภาพ CT ของต้นบอนไซเมื่อใช้ฟังก์ชันถ่ายโอนแตกต่างกัน

5.2.2 การเปลี่ยนการให้แสงเงา

การทดสอบผลของการเปลี่ยนการให้แสงเงาทำได้โดยการเลือกชุดข้อมูลเชิงปริมาตรจำนวน 3 ชุดแล้วทดลองสร้างภาพเชิงปริมาตรโดยกำหนดให้มีการให้แสงเงาที่แตกต่างกันคือ แบบปกติไม่มีการให้แสงเงา แบบให้แสงเงาแบบแสงแพร่ และแบบให้แสงเงาแบบแสงกล้า

(ก) แบบปกติ *(ข) แบบให้แสงเงาแบบแสงแพร่* *(ค)แบบให้แสงเงาแบบแสงกล้า*

รูปที่ 5.5 ภาพภายในช่องหูเมื่อใช้การให้แสงเงาแบบต่าง ๆ



(ก) แบบปกติ *(ข) แบบให้แสงเงาแบบแสงแพร่* *(ค)แบบให้แสงเงาแบบแสงกล้า*

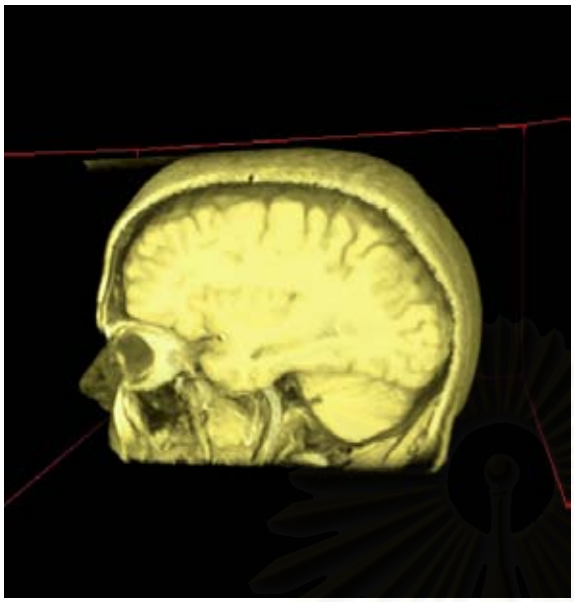
รูปที่ 5.7 ภาพโมเลกุลของโปรตีนเมื่อใช้การให้แสงเงาแบบต่าง ๆ

(ก) แบบปกติ *(ข) แบบให้แสงเงาแบบแสงแพร่* *(ค)แบบให้แสงเงาแบบแสงกล้า*

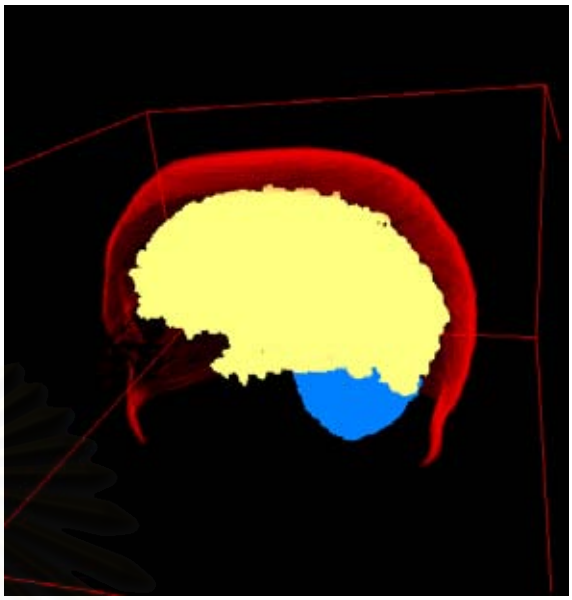
รูปที่ 5.8 ภาพ MR ของศีรษะเมื่อใช้การให้แสงเงาแบบต่าง ๆ

5.2.3 การใช้ข้อมูลค่าสเกลาร์และค่าฉลาก

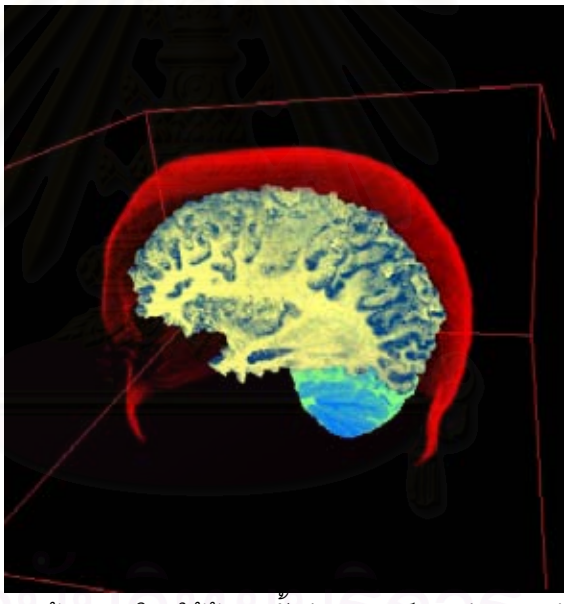
ในการทดลองการสร้างภาพโดยใช้ข้อมูลแบบต่าง ๆ คือ ใช้ข้อมูลเฉพาะค่าสเกลาร์ ใช้ข้อมูลเฉพาะค่าฉลาก และใช้ทั้งข้อมูลค่าสเกลาร์และค่าฉลากในการกำหนดสีของแต่ละจุดบนภาพทำได้โดยการเลือกชุดข้อมูลที่มีทั้งค่าสเกลาร์และค่าฉลากมาหนึ่งชุดแล้วทำการสร้างภาพจากชุดข้อมูลนั้น โดยใช้ข้อมูลนำเข้าแบบต่าง ๆ ผลที่ได้ปรากฏอยู่ในรูปที่ 5.9 ซึ่งแสดงให้เห็นว่า การใช้ข้อมูลค่าสเกลาร์อย่างเดียวในรูปที่ 5.9 ก นั้นจะสามารถเห็นรายละเอียดของวัตถุที่ต้องการสร้างภาพได้ (ขึ้นอยู่กับฟังก์ชันถ่ายโอนที่ใช้) แต่ไม่สามารถกำหนดให้แสดงเฉพาะส่วนที่ต้องการได้ โดยเฉพาะในกรณีที่ค่าสเกลาร์ของส่วนที่ต้องการและไม่ต้องการแสดงมีค่าใกล้เคียงกันสีที่ได้จะมีลักษณะคล้ายคลึงกัน ในทางตรงกันข้ามการสร้างภาพโดยใช้เพียงค่าฉลากอย่างเดียวในรูปที่ 5.9 ข สามารถเลือกเฉพาะส่วนที่ต้องการมาแสดงได้ และสามารถเลือกกำหนดสีที่ให้กับส่วนต่าง ๆ ได้ แต่จะไม่เห็นรายละเอียดภายในส่วนนั้นนั้นเนื่องจากถูกแทนที่ด้วยสีเดียวกัน และการสร้างภาพโดยใช้ทั้งข้อมูลค่าสเกลาร์และค่าฉลากร่วมกันในรูปที่ 5.9 ค นั้นทำให้ได้ข้อดีของทั้งสองแบบคือ สามารถเลือกให้แสดงเฉพาะส่วนที่ต้องการได้ สามารถกำหนดโทนสีให้กับส่วนต่าง ๆ ได้ และสามารถเห็นรายละเอียดภายในส่วนนั้นตามที่กำหนดโทนสีไว้



(ก) การสร้างภาพโดยใช้ข้อมูลค่าสเกลาร์อย่างเดียว



(ข) การสร้างภาพโดยใช้ข้อมูลค่าฉลากอย่างเดียว



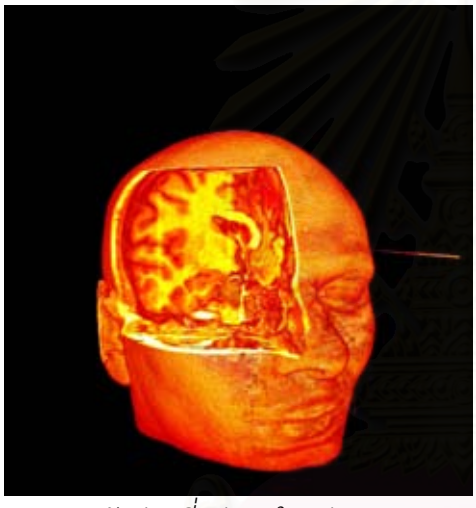
(ค) การสร้างภาพโดยใช้ข้อมูลทั้งค่าสเกลาร์และค่าฉลากร่วมกัน

รูปที่ 5.9 การสร้างภาพเชิงปริมาตรจากข้อมูลนำเข้าแบบต่าง ๆ

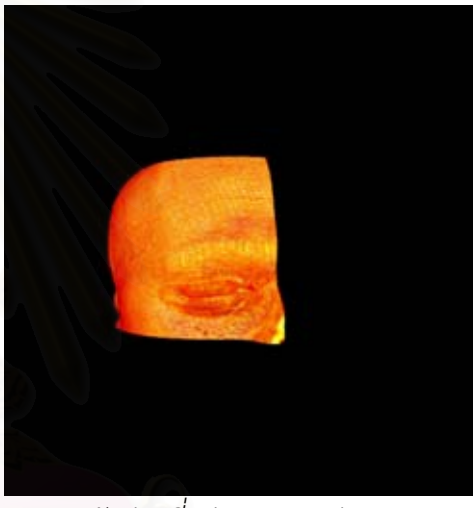
5.2.4 การตัด

การทดสอบการตัดทำได้โดยการเลือกชุดข้อมูลเชิงปริมาตรขึ้นมาหนึ่งทำการสร้างภาพซึ่งถูกตัดด้วยการตัดแบบต่าง ๆ คือ การตัดโดยใช้ระนาบ (รูปที่ 5.10) การตัดโดยใช้รูปทรงลูกบาศก์ (รูปที่ 5.11) การตัดโดยใช้รูปทรงกลม (รูปที่ 5.12) การตัดโดยใช้รูปทรงกระบอก (รูปที่ 5.13) และการตัดโดยใช้รูปทรงปริซึม (รูปที่ 5.14)

รูปที่ 5.10 การตัดโดยใช้ระนาบ

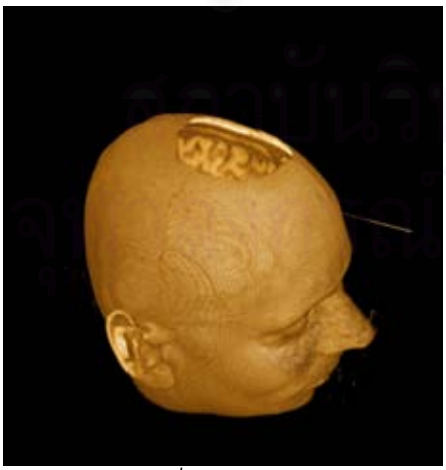


(ก) ตัดส่วนที่อยู่ภายในรูปทรงออก



(ข) ตัดส่วนที่อยู่ภายนอกรูปทรงออก

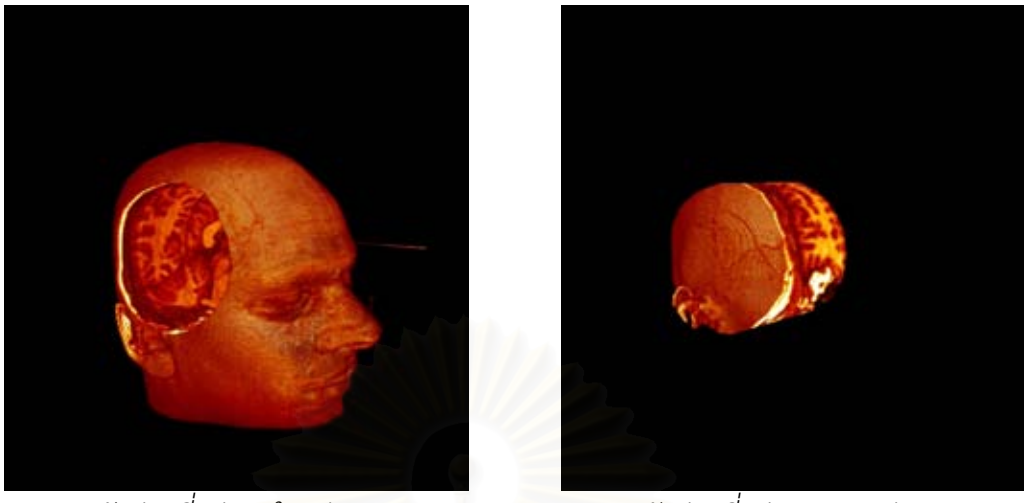
รูปที่ 5.11 การตัดโดยใช้รูปทรงสี่เหลี่ยม



(ก) ตัดส่วนที่อยู่ภายในรูปทรงออก

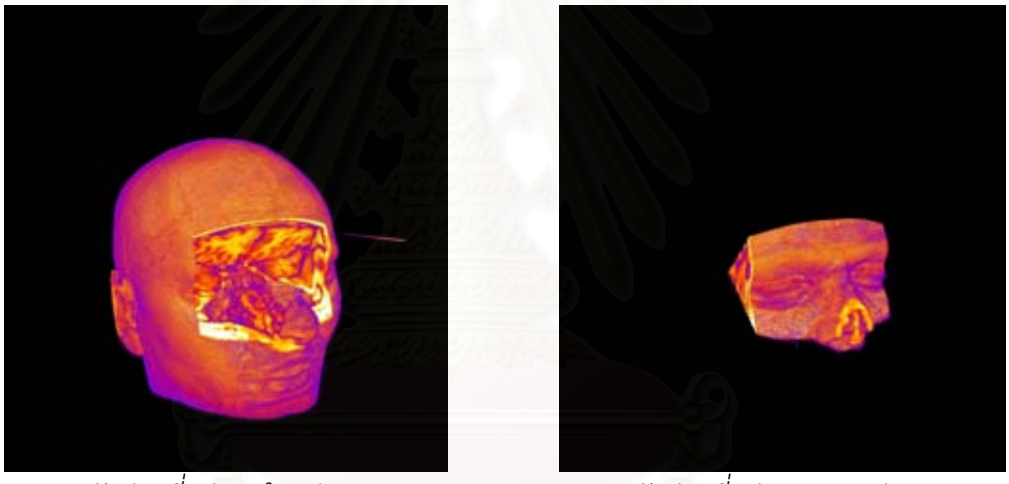
(ข) ตัดส่วนที่อยู่ภายนอกรูปทรงออก

รูปที่ 5.12 การตัดโดยใช้รูปทรงกลม



(ก) ตัดส่วนที่อยู่ภายในรูปทรงออก *(ข) ตัดส่วนที่อยู่ภายนอกรูปทรงออก*

รูปที่ 5.13 การตัดโดยใช้รูปทรงกระบอก



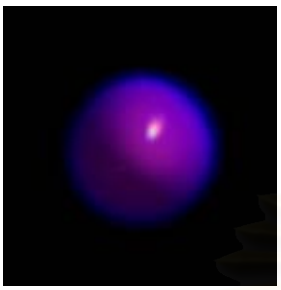
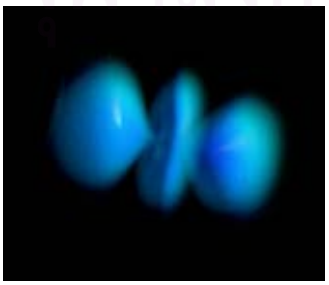
(ก) ตัดส่วนที่อยู่ภายในรูปทรงออก *(ข) ตัดส่วนที่อยู่ภายนอกรูปทรงออก*

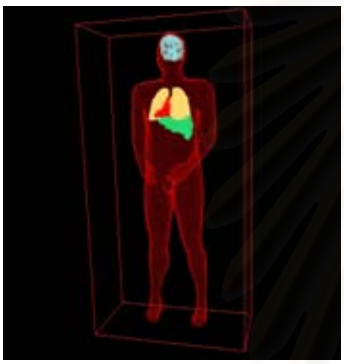
รูปที่ 5.14 การตัดโดยใช้รูปทรงปริซึม

5.3 ประสิทธิภาพในการสร้างภาพ

ประสิทธิภาพในการสร้างภาพของส่วนโปรแกรมสามารถวัดได้จากการวัดเวลาที่ใช้ในการสร้างภาพของส่วนโปรแกรมโดยมีหน่วยเป็นมิลลิวินาที ในการทดลองนั้นได้ใช้เครื่องคอมพิวเตอร์ส่วนบุคคล ที่มีประมวลผลกลางคือ Intel Pentium4 1.8 GHz หน่วยความจำหลักขนาด 512 เมกะไบต์ ทำทดลอง 2 ครั้งคือ ครั้งแรกใช้กราฟิกส์ฮาร์ดแวร์ เอ็นวิเดีย จีฟอร์ซ 4 4600 หน่วยความจำ 128 เมกะไบต์ และครั้งที่ 2 เปลี่ยนกราฟิกส์ฮาร์ดแวร์เป็น เอทีไอ ราเดออน 8500 หน่วยความจำ 64 เมกะไบต์ โดยแต่ละครั้งจะทำการทดลองกับข้อมูลเชิงปริมาตรที่มีขนาดต่าง ๆ กันจำนวน 8 ชุดข้อมูล และทดลองสร้างภาพทั้งแบบไม่มีการให้แสงเงา และมีการให้แสงเงาแบบแสงกล้าได้ผลดังตารางที่ 5.1

ตารางที่ 5.1 ประสิทธิภาพในการสร้างภาพของส่วนโปรแกรม

ข้อมูลเชิงปริมาตร (ขนาด)	ไม่มีการให้แสงเงา (ms)		ให้แสงเงาแบบแสงกล้า (ms)	
	เอ็นวิเดีย	เอทีไอ	เอ็นวิเดีย	เอทีไอ
 แบบจำลองนิวคลีออน (41x41x41)	24	23	35	55
 แบบจำลองการฉีดเชื้อเพลิง (64x64x64)	24	23	35	61
 ภาพภายในช่องหู (128x128x30)	35	27	59	122
 แบบจำลองอะตอมของไฮโดรเจน (128x128x128)	35	32	70	119

ข้อมูลเชิงปริมาตร (ขนาด)	ไม่มีการให้แสงเงา (ms)		ให้แสงเงาแบบแสงกล้า (ms)	
	เอ็นวิเดีย	เอทีไอ	เอ็นวิเดีย	เอทีไอ
 ภาพ CT ของศีรษะ (256x256x106)	47	77	118	256
 ภาพค่าฉลากของมนุษย์เต็มตัว (493x87x147)	47	266	118	มากกว่า 10 วินาที
 ภาพ MR ของศีรษะ (256x256x256)	47	385	154	มากกว่า 10 วินาที
 ภาพ CT ของเท้า (256x256x256)	47	357	141	มากกว่า 10 วินาที

จากตารางที่ 5.1 สังเกตได้ว่าประสิทธิภาพในการสร้างภาพของส่วนโปรแกรมเมื่อทำงานบนกราฟิกส์ฮาร์ดแวร์ของเอ็นวิเดียสูงกว่าเมื่อทำงานบนกราฟิกส์ของเอทีไอยกเว้นในกรณีที่ขนาดของชุดข้อมูลมีขนาดเล็กและไม่มีการให้แสงเงา การให้แสงเงาส่งผลให้ประสิทธิภาพในการสร้างภาพลดลงค่อนข้างมากโดยเฉพาะในชุดข้อมูลขนาดใหญ่ อย่างไรก็ตามแม้บนชุดข้อมูลขนาดใหญ่ที่สุดที่ได้ทดสอบ (256x256x256) ประสิทธิภาพในการสร้างภาพยังสูงเพียงพอที่จะโต้ตอบกับผู้ใช้ได้อย่างรวดเร็ว ยกเว้นในกรณีที่ทำงานบนเอทีไอกราฟิกส์ฮาร์ดแวร์และมีการให้แสงเงาเนื่องจากหน่วยความจำของกราฟิกส์ฮาร์ดแวร์ไม่เพียงพอที่จะเก็บเท็กซ์เจอร์ของเวกเตอร์ปกติ กราฟิกส์ฮาร์ดแวร์จึงต้องนำเท็กซ์เจอร์ไปเก็บไว้บนหน่วยความจำหลักของเครื่องส่งผลให้ประสิทธิภาพลดลงอย่างมาก

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

จากการทดลองและวิเคราะห์ผลที่ผ่านมา สามารถสรุปผลการวิจัยและข้อเสนอแนะเพื่อเป็นแนวทางในการพัฒนาต่อไปในอนาคตได้ ดังนี้

6.1 สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อออกแบบและพัฒนาชุดส่วนโปรแกรมเพื่อใช้ในการสร้างภาพเชิงปริมาตรในด้านการแพทย์ งานวิจัยนี้แบ่งออกเป็นสององค์ประกอบหลักคือ การออกแบบส่วนโปรแกรมแต่ละส่วนภายในชุดส่วนโปรแกรม และขั้นตอนวิธีที่ใช้ในการสร้างภาพ

ในส่วนของการออกแบบส่วนโปรแกรมนั้นได้ออกแบบให้แต่ละส่วนโปรแกรมแยกอิสระไม่ขึ้นต่อกัน แต่ใช้รูปแบบข้อมูลนำเข้าและส่งออกร่วมกัน ทำให้เกิดข้อดีในการนำส่วนโปรแกรมไปใช้งานโดยผู้ใช้สามารถเลือกเฉพาะส่วนที่ต้องการได้ อีกทั้งรูปแบบของข้อมูลนำเข้าและส่งออกยังอยู่ในรูปแบบที่ง่ายการต่อขยายจึงสามารถทำได้อย่างไม่ยุ่งยาก

ในส่วนขั้นตอนวิธีที่ใช้ในการสร้างภาพงานวิจัยนี้เลือกใช้ขั้นตอนวิธีที่ใช้กราฟิกส์ฮาร์ดแวร์เข้ามาช่วยเร่งความเร็วในการสร้างภาพ เนื่องจากต้องการลดเวลาที่ใช้ในการสร้างภาพเชิงปริมาตรซึ่งปกติใช้เวลานาน นอกจากนี้ยังเพิ่มความสามารถในการให้แสงเงา และการตัดโดยใช้ระนาบหรือรูปทรง ซึ่งทั้งหมดถูกกระทำบนกราฟิกส์ฮาร์ดแวร์

จากผลการทดลองแสดงให้เห็นว่าชุดส่วนโปรแกรมที่สร้างขึ้นสามารถนำไปสร้างโปรแกรมประยุกต์สำหรับการสร้างภาพเชิงปริมาตรได้ และมีความยืดหยุ่นเพียงพอที่จะทำงานบนอินเตอร์เน็ตในรูปแบบของเว็บเพจ ประสิทธิภาพในการสร้างภาพที่ได้อยู่ในเกณฑ์ดีโดยเวลาที่ใช้ในการสร้างภาพที่น้อยที่สุดอยู่ที่ 23 มิลลิวินาที แม้ว่าการให้แสงเงาและขนาดของข้อมูลเชิงปริมาตรที่ใหญ่ขึ้นจะส่งผลกระทบเวลาที่ใช้ในการสร้างภาพเพิ่มขึ้นแต่ก็ยังอยู่ในระดับที่สามารถโต้ตอบกับผู้ใช้ได้อย่างรวดเร็วโดยเวลาที่มากที่สุดที่ใช้ในการสร้างภาพอยู่ที่ 385 มิลลิวินาที ยกเว้นในกรณีที่หน่วยความจำของกราฟิกส์ฮาร์ดแวร์ไม่สามารถบรรจุข้อมูลเชิงปริมาตรได้ทั้งหมดประสิทธิภาพที่ได้จะต่ำมาก

6. 2 ข้อสังเกตและข้อเสนอแนะ

จากผลการวิจัยโดยรวมนั้นพบว่างานวิจัยนี้ยังมีส่วนต่าง ๆ ที่ควรต้องพัฒนาเพิ่มเติมและปรับปรุง ดังนี้

1. เพิ่มความสามารถให้ผู้ใช้สามารถนำวัตถุไปแสดงร่วมกับการสร้างภาพเชิงปริมาตรได้
2. ปรับปรุงให้ผู้ใช้สามารถปรับค่าดัชนีสะท้อนแสงของการให้แสงเงาแบบแสงกล้าได้
3. ปรับปรุงให้ผู้ใช้สามารถปรับสีของแสงแพร่และแสงกล้าได้
4. เพิ่มคุณภาพของภาพที่ได้โดยใช้ตัวกรองการสร้างใหม่อันดับสูงขึ้น
5. เพิ่มความสามารถในการสร้างภาพสามมิติแบบสเตอริโอ

รายการอ้างอิง

1. Kitware, Inc. The Visualizationi Toolkit. U.S.A. Available from: <http://www.vtk.org/>, 2001.
2. Silicon Graphics Inc. Volumizer, U.S.A. Available from <http://www.sgi.com/software/volumizer/>, 2002.
3. Aicom Software Limited Volumaid U.K. Available from: <http://www.volumaid.com/>, 2001.
4. Alan Watt, 3D Computer Graphics Second Edition, Great Britain: Addison-Wesley Publishing Company, 1993.
5. Robert A. Drebin, Loren Carpenter and Pat Hanrahan. Volume Rendering, Computer Graphics, vol. 22, Number 4, August 1988.
6. Mark Segal, Kurt Akeley. The OpenGL Graphics System: A Specifications (Version 1.2.1), Available from: <http://www.opengl.org/>, 1999.
7. NVIDIA Corporation. NVIDIA OpenGL Extension Specifications. Available from: <http://www.nvidia.com/>, 2003.
8. ATI Technologies Inc. ATI OpenGL Extension Support. Available from: <http://www.ati.com/>, 2003.
9. Allen Van Gelder, Kwansik Kim. Direct Volume Rendering with Shading via Three-Dimensional Textures. Proceedings of the 1996 symposium on Volume visualization, pp.23-ff, San Francisco, California, United States, 1996.
10. Klaus Engel, Martin Kraus, Thomas Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware, pp.9-16, Los Angeles, California, United States, 2001.
11. B.T. Phong. Illumination for Computer Generated Pictures. (n.p.): Communications of the ACM, 18(6):311-317, 1975.
12. Irwin Sobel. An Isotropic 3x3x3 Volume Gradient Operator. (n.p.): Technical report, Hewlett-Packard Laboratories, 1995.
13. S. W. Zucker and R. A. Hummel. A Three-Dimensional Edge Operator. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-3(3):324-331, May 1981.
14. David Chappell. Understanding ActiveX and OLE, U.S.A.: Microsoft Press, 1996.

15. Manson Woo, Jackie Neider, Tom Davis, and Dave Shreiner. OpenGL Programming Guide Third Edition. U.S.A: Addison Wesley Longman, 1999.

16. M. Meißner, U. Hoffmann, and W. Straßer. Enabling Classification and Shading for 3D Texture Mapping based Volume Rendering using OpenGL and Extensions. In Proc. of IEEE Visualization, pages 207–214, San Franisco, CA, USA, October 1999. IEEE Computer Society Press.

17. F. Dachille, K. Kreeger, B. Chen, I. Bitter, and A. Kaufman. High-Quality Volume Rendering Using Texture Mapping Hardware. In Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware, pages 69–76, Lisboa, Portugal, August 1998.

18. R. Westermann and T. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Application. In Computer Graphics, Proc. Of ACM SIGGRAPH,pages 169-177, Orlando, FL,U.S.A, August 1998.

19. C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive volume rendering on standard pc graphics hardware using multi-texturing and multi-stage rasterization. In Proc. of Eurographics/SIGGRAPH Workshop on Graphics Hardware, pages 109–118, Interlaken, Switzerland, August 2000.

20. Michael Meißner and Stefan Guthe and Wolfgang Straßer. Higher Quality Volume Rendering on PC Graphics Hardware, Wilhelm Schickard Institute for Computer Science, Graphical-Interactive Systems (WSI/GRIS), University of Tübingen, 2001.

21. สุพจน์ จันทร์วิพัฒน์. การสร้างภาพ 3 มิติ จากลำดับของข้อมูลภาพตัดขวางโดยใช้วิธีพิจารณาจากเวคเตอร์ปกติของพื้นผิว. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2541.

22. วรเทพ ไพบูลย์รัตนากร. การเพิ่มความเร็วให้การสร้างภาพเชิงปริมาตรทางการแพทย์โดยใช้การแปลงระยะทาง. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2544.

23. เกษมสุข เสพศิริสุข. การสร้างภาพเชิงปริมาตรทางการแพทย์แบบเร็วโดยใช้การแปลงเฉือนและบิด. วิทยานิพนธ์ปริญญามหาบัณฑิต สาขาวิชาวิศวกรรมอิเล็กโทรนิกส์ บัณฑิตวิทยาลัย, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2544.

24. ชุติมา เกษมกรกิจ, และเชาวน์ยุทธ ขจรศักดิ์ว่องไว. เทคนิคและเครื่องมือประกอบภาพ 3 มิติจากชุดภาพตัดขวาง 2 มิติ. โครงงานทางวิศวกรรม สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2543.

ภาคผนวก

ภาคผนวก ก

บทความที่นำเสนอในงานการประชุมวิชาการ

บทความเรื่อง “Programmable Web-based Volume Visualization” นำเสนอในงานประชุมวิชาการ The First Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology (ECTI) Annual Conference จัดที่โรงแรม Amari Orchid Resort พัทยา จังหวัดชลบุรี ระหว่างวันที่ 13-14 มีนาคม พ.ศ. 2547 บทความนี้ตีพิมพ์ไว้ใน The First Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology (ECTI) Annual Conference หน้า 165-168

Programmable Web-based Volume Visualization

Sakpod Tongleamnak[1], Nongluk Covavisaruch[2], and Wiwat Vatanawood[3]

Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand.
sakpod.t@student.chula.ac.th[1], nongluk.c@chula.ac.th[2], and wiwat@chula.ac.th[3]

ABSTRACT

This paper proposes an approach and a prototype of programmable volume visualization in web environment. The approach is based on client-side visualization and 3D texture volume rendering. Programmable visualization is created by separating rendering component from user interface and making interface between rendering component and web browser script language. This allows programmers to write the script to control the visualization process. From our prototype implementation, it is found that client-side programmable web-based volume visualization approach can achieve good performance.

Keywords: Volume Visualization, Web-based Visualization, Texture Mapping.

1. INTRODUCTION

Volume visualization is an effective technique for analysis and for presenting volume data. The volume data can be acquired either from measuring with some equipments (such as CT - Computer Tomography or MR - Magnetic Resonance images) or from computing in many fields of work such as medical, science, and engineering. Many approaches have been proposed and developed to visualize volume data. They can be categorised into two groups: software approach and hardware approach. The software approach is usually limited in terms of performance due to a large amount of data and high computation. Some hardware approaches are good in performance but their specific hardware is too expensive and they lack of the availability to use in a widespread system. When 3D graphics hardware on PC became powerful, it was recognized to be useful in volume visualization [1] by using its texture mapping capability to gain the performance of visualization. This approach is proved to have high performance enough to achieve interactive frame rate even on moderate cost PCs. Moreover, with a catalyst of game business at present, the 3D graphics hardware is now becoming more available, more powerful and cheaper (less than $100 for moderate class). Therefore, 3D graphics hardware is now an attractive choice to utilize in real-life applications.

Nowadays, information can be accessed from everywhere through the internet but volume visualization is still limited on this environment. The reason that usually be addressed is the lacking of computational performance on client machines. To our knowledge, there are several works related to volume visualization in web environment. Isosurface extraction approaches as proposed in [2] try to extract surface from the volume data on a server and use VRML to display the result surface on client. By avoiding high computation in volume visualization, these approaches get acceptable to high performance but the extraction cause data lost which can affect some applications such as in the medical area. Another VRML approach as in [3] uses texture mapping technique to visualize volume data without extracting surface but the limitation of VRML makes this approach unable to use graphics hardware to render shaded image in volume visualization.

Dynamic web pages can change their page's structure, style, or content after the page is loaded in the browser without having to request data from a web server. Our approach to build programmable visualization is by creating visualization applications in dynamic web pages. Advantage of this approach is not only that we can program the visualization but also that bandwidth is saved, and response time is short, while users are allowed to manipulate their visualization.

In this paper, we describe our conceptual design approach and theoretical background in section 2. Our prototype implementation's results are discussed in section 3. Finally, in section 4 we conclude our paper and outline the future work.

2. DESIGN APPROACH

The objective of our work is to create a programmable visualization that has good performance to achieve interactive frame rate. Therefore, our design is based on two approaches: a client-side visualization (section 2.1) and a volume rendering via 3D texture mapping (section 2.2). The former is a framework to create a web-based visualization and the latter is an algorithm to render volume datasets. The client-side visualization requires that users allow web browser to install visualization software on client machines. The algorithm requires that client machines equip with 3D graphics hardware. In order to improve the appearance of the result images, we add shading (section 2.3) in our approach. Section 2.4 describes our technique to add programmable ability to the visualization.

2.1 Client-Side Visualization

In the client-side visualization approach, most of computation processes are executed on the client machine. Unlike the server-side visualization, performance does not depend on number of simultaneous

client machines which are very limited in high computational visualization like volume visualization. The client-side visualization approach assumes that visualization software is already installed on the client machine or downloaded from server to the client machine. A volume data can be downloaded from server or read from local file. Finally, the client machine executes the visualization software to render visual image of the volume data.

In the case that volume datasets are not on the client machine, we have to transfer it from server to client. Since the major drawback of client-side visualization is the transferring of typical large volume datasets from server to client through the internet, we suggest that datasets be compressed before transferring (Fig. 1). As volume datasets are actually binary data, thus all lossless binary compression algorithm (i.e. Huffman or Arithmetic coding) can be used. After data transfer, the client decompresses the datasets and uses it in rendering process.

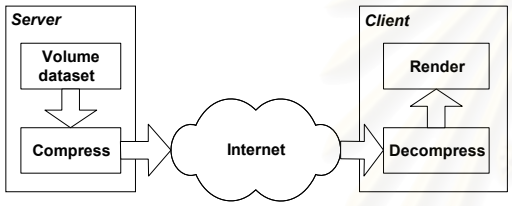
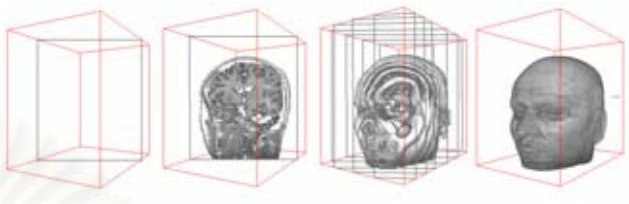


Fig.1: *Data Transfer*

2.2 Volume Rendering via 3D Texture Mapping

Texture mapping is ability of graphics hardware to draw textures on 3D surfaces. Textures are simply arrays of data - for example color data, or luminance data. The individual values in a texture array are called *texels*. Types of textures are categorized by number of dimensions of texture array. Capability of using 3D textures is a part of OpenGL 1.2 standard [4] which is supported by most current consumer PC's graphics hardware (e.g. NVIDIA GeForce3, 4, FX, ATI Radeon). This capability is proved that can be useful in volume rendering [5]. The main idea is to use texture mapping as trilinear interpolation by interpreting datasets to 3D texture. Since both volume datasets and textures are array of data, interpreting is done by simply convert voxel data to texel data. We render the dataset by applying the texture to multiple planes parallel to the image plane, called *resampling slices*. These slices are also clipped against the texture domain and sent to the geometry processing unit of graphics hardware in back-to-front order. In rasterization state, the hardware divides each resampling slice into many fragments. A value of each fragment is calculated by graphics hardware using trilinear interpolation within the texture for reconstructing the texture. Finally, in order to approximate continuous volume rendering integral we use hardware alpha blending to compose pixel's value in frame buffer with the fragment value by using opacity-weighted colours. Overall of the process is shown in Fig 2; "a" is one resampling slice and texture domain boundary; "b" is after hardware reconstruct the volume texture on the slice; "c" is reconstruction of many slices; "d" is final image after composing all resampling slices. Since all of high computational processes like interpolation and composing are done in the hardware, this approach is proved that in a moderate data size (256^3) interactive frame rate can be archived [1].



(a) *(b)* *(c)* *(d)*

Fig.2: *Volume Rendering via 3D Texture*

2.3 Shading

Shading is an interesting feature in 3D visualization which has to present 3D model in 2D image, because it makes the visualization more dimensional and more realistic. Figure 3 shows the difference between with and without shading in volume visualization. Phong illumination model [7] is the most popular model for shading because it can produce good image with uncomplicated computation. The model computes shading as a summation of three different terms which are an *ambient* term, a *diffuse* term, and a *specular* term. The calculation of Phong illumination model is show in equation 1-4.

$$I_{Phong} = I_{ambient} + I_{diffuse} + I_{specular} \quad (1)$$

$$I_{ambient} = k_a \quad (2)$$

$$I_{diffuse} = I_p k_d \left(\vec{l} \bullet \vec{n}\right) \quad (3)$$

$$I_{specular} = I_p k_s \left(\vec{h} \bullet \vec{n}\right)^n \quad (4)$$

I_{Phong}, $I_{ambient}$, $I_{diffuse}$, and $I_{specular}$ are intensity of each term. k_a, k_d, and k_s are material property determines amount of reflection in each term. $\vec{l}$ is light source direction. $\vec{n}$ is normal vector of surface. $\vec{h}$ is halfway vector of light source direction and viewing direction. The exponent n is shininess of the surface.

(a) *(b)*

Fig.3: *(a) Volume Rendering without Shading and (b) Volume Rendering with Shading*

Since vertices of geometry in 3D texture mapping approach is only at the corner of resampling slices, per-vertex shading of OpenGL cannot be used. To achieve per-fragment (per-pixel) shading we use OpenGL extension called *fragment shader* (pixel shader). OpenGL multi-texturing allows programmer to map more than one texture to surface at a time and color of textures on surface is combined by *texture environment* which is a set of few inflexible functions. The extension allows programmer to program the combination of texture's values in graphics hardware with more powerful functions like addition, multiplication and dot product. Due to all of operators in equation 1-4 are included in the extension (the exponent n can be complete by multiple multiplications); the shading is proved to be computed in graphics hardware by using the extension [8]. Since volume datasets doesn't have real normal vector which is necessary to compute the shading, we use normalized gradient vector of the scalar in datasets instead. The vector is sent to graphics hardware by converting magnitude of vector in each axis to intensity of each RGB channel and storing it in a 3D texture. The normal texture is mapped to resampling slices at the same coordinates of scalar texture. Finally, we program the fragment shader state to compute the shading in equation 1-4. While the hardware reconstructing volume on resampling slices each fragment contains both scalar value and normal vector of that position. Fragment shader uses this vector with input light direction and viewing direction to compute the shaded value of fragment.

2.4 Programmable Visualization Technique

Dynamic web pages are programmed by a set of instructions called script. There are two types of scripts according to where they are executed, client-script and server-script. Since we use client approach, our script is client-script. Client-script is downloaded to the client's web browser as part of the HTML document. The client's web browser interprets and executes the script. Programmers write script in script language like VBScript and JavaScript to describe how web browser should handle events that come from user through controls such as buttons, text boxes, list boxes; or events on changing status of browser such as on load, on close, and on size. We separate rendering component from user interface and make interface between the component and script languages (see Fig. 4). The interface is composed of many methods and properties that can manipulate visualization like changing view port, transfer function, light direction. Programmers can create their own user interface and use the languages to describe how visualization should manipulate according to events that occur. This also could be useful in consultancy works. The consultants can create a script that describe their visualization; user just open the page then the visualization application execute. Furthermore rendering component can be reused in different page, different user interface, and different content. A limitation of client-script is that it cannot be run without supported web browser.

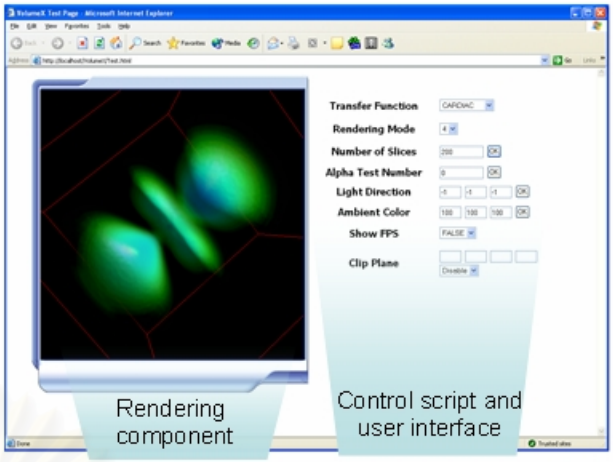


Fig.4: *Programmable Visualization*

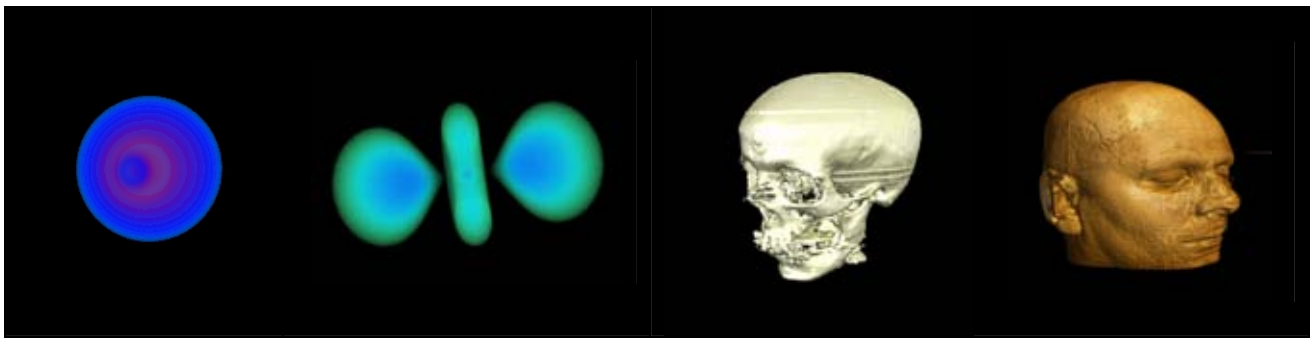
3. RESULT AND DISCUSSION

Our prototype is composed of 2 parts: the rendering component implemented in C++ and distributed in a form of ActiveX control. The control script is written in VBScript. Interface of the control is composed of 3 groups of methods and properties according to their functional:

- Scene Manipulation (object orientation, light direction, scale, background color, etc.)
- Buffer Management (loading volume data to rendering core's buffer, deleting data in rendering core's buffer, etc.)
- Utility (capturing result image, show frame per second, etc.)

The component uses rendering hardware through OpenGL and extensions. Both server and client are standard PC but the client is equipped with a NVIDIA GeForce4 Ti4600 graphics card; they are linked together via 100Mbps Ethernet connection. The client web browser is the Internet Explorer 6.0 which already supports VBScript. We use Huffman algorithm in compression process. Table 1 shows the sizes of datasets before and after compression, compress ratio and transferring time. Performance of the prototype in rendering different size of datasets to 500×500 pixels view port with and without shading are shown in table 2. Figure 5 shows the result images of each dataset.

The compression ratios of Huffman compression (Table 1) only depend on redundancy of the datasets without associate with size or complexity. Due to high speed connection of the prototype, the expected drawback in consuming transferred time (Table 1) is merely noticed but approximate calculation from amount of data of Human Head MR dataset, it will take more than 32 minutes to transfer through 56Kbps Modem. Performance of the prototype (Table 2) is varying to size of datasets and shading approximately takes twice computation time than non-shaded rendering.



(a) Non-Shaded Nucleon *(b) Non-Shaded Hydrogen Atom* *(c) Shaded Human Head CT* *(d) Shaded Human Head MR*

Fig.5: *Result Images*

Table 1: *Result of Huffman Compression and Transfer Time*

Dataset	Compression			Time (s)
	Size Before (byte)	Size After (byte)	Ratio	
Nucleon	68,921	53,661	1.28	0.07
Hydrogen	2,097,152	632,890	3.31	0.23
Human Head CT	6,946,816	2,322,373	2.99	0.57
Human Head MR	16,777,216	13,834,186	1.21	6.51

Table 2: *Rendering Performance*

Dataset	Size (voxel)	Frame Per Second	
		No Shade	Shaded
Nucleon	41x41x41	75	37
Hydrogen	128x128x128	37	25
Human Head CT	256x256x106	18	8
Human Head MR	256x256x256	11	5

4. CONCLUSION AND FUTURE WORKS

We have presented approach to implement volume rendering applications in web environment. Our prototype performance is high enough to achieve interactive frame rate while rendering moderate size datasets. Furthermore, programmable visualization allows programmers to prepare both automatic and users control visualization. Even after compression, most datasets are still large enough to cause drawback in low speed connection. In future works we will investigate new data compression or transfer method to reduce effect of the drawback.

5. REFERENCE

[1] C. Rezk-Salama, K. Engel, G. Greiner, T. Ertl, "Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization" *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pp.109-118, Interlaken Switzerland, 2000.

[2] Klaus D. Engel, Rüdiger Westermann, and Thomas Ertl, "Iso-surface extraction techniques for web-based volume visualization", *IEEE Visualization '99*, pp.139-146, San Francisco, 1999.

[3] Johannes Behr, Marc Alexa, "Volume Visualization in VRML", *Proceedings of the sixth international conference on 3D Web technology*, pp.23-27, Paderbon, Germany, 2001.

[4] Mark Segal, Kurt Akeley, "The OpenGL Graphics System: A Specifications (Version 1.2.1)", http://www.opengl.org, 1999.

[5] Allen Van Gelder, Kwansik Kim, "Direct Volume Rendering with Shading via Three-Dimensional Textures", *Proceedings of the 1996 symposium on Volume visualization*, pp.23-ff, San Francisco, California, United States, 1996.

[6] NVIDIA Corporation, "NVIDIA OpenGL Extension Specifications", http://www.nvidia.com, 2003.

[7] B.T. Phong, "Illumination for Computer Generated Pictures", *Communications of the ACM*, 18(6):311-317, 1975.

[8] Klaus Engel, Martin Kraus, Thomas Ertl, "High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading", *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pp.9-16, Los Angeles, California, United States, 2001.

ประวัติผู้เขียนวิทยานิพนธ์

นายศักดิ์พจน์ ทองเลี่ยมนาค เกิดวันที่ 8 มิถุนายน พ.ศ. 2522 ที่จังหวัดชัยภูมิ สำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิตจากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง หลังจากนั้นได้เข้ามาศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2544