การพัฒนาโปรแกรมจำลองแรงสัมผัสของการวาดภาพด้วยสีเทียนโดยใช้อุปกรณ์จำลองการสัมผัส

Development of a force-reflective simulation program for wax crayon drawing using a haptic force feedback device

โดย

นายธีรเดช โอทกานนท์

รหัส 45066002

อาจารย์ที่ปรึกษา

รศ.ดร.นพพร โชติกกำธร

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 1 ปีการศึกษา 2547

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อเรื่อง	การพัฒนาโปรแกรมจำลองแรงสัมผัสของการวาดภาพด้วยสีเทียนโดยใช้อุปกรณ์จำลองการสัมผัส
นักศึกษา	นายธีรเดช โอทกานนท์
อาจารย์ที่ปรึกษา	รศ.ดร.นพพร โชติกกำธร
ระดับการศึกษา	วิทยาศาตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2547

บทคัดย่อ

การพัฒนาระบบสภาพแวดล้อมเสมือนโดยใช้อุปกรณ์จำลองการสัมผัสในการจำลองการวาดภาพด้วยสีเทียน ประกอบไปด้วย 2 ส่วนหลัก กล่าวคือ ส่วนแรกจะเป็นการพัฒนาในส่วนของกราฟิกซึ่งใช้ในการแสดงผลรูปภาพของกระดาษและรูปภาพที่ได้ภายหลังจากที่ได้ทำการวาดภาพ อีกส่วนหนึ่งเป็นการจำลองแรงสัมผัสซึ่งแรงสัมผัสประกอบไปด้วย 2 แรง คือแรงต้านการเคลื่อนที่เป็นแรงที่จำลองแรงเสียดทานในการวาดภาพ และอีกแรงหนึ่งคือแรงที่จำลองลักษณะของพื้นผิวของกระดาษ เพื่อให้ผู้ใช้สามารถรับรู้ถึงความขรุขระของพื้นผิวกระดาษ ให้เกิดความสมจริงในการวาดภาพในสภาพแวดล้อมแบบเสมือน ในการพัฒนานี้จะพัฒนาในส่วนของการจำลองแรงสัมผัสของการวาดภาพ โดยจะทำการอ่านค่าความสว่างของรูปภาพของกระดาษจำลอง มาจำลองเป็นลักษณะของความขรุขระของพื้นผิวของกระดาษ และอีกส่วนหนึ่งนั้นเป็นการจำลองแรงต้านการเคลื่อนที่ซึ่งจะกำหนดค่าสัมประสิทธิ์แรงเสียดทานในการต้านการเคลื่อนที่ในโครงการนี้กำหนดให้สัมประสิทธิ์แรงเสียดทานของกระดาษ คือ 0.3 และของสีเทียนคือ 4.0 ส่วนการจำลองกระดาษใช้วิธีการของ texture mapping ในการแมปรูปภาพของกระดาษลงบนโพลิกอน และการแสดงลายเส้นของการวาดภาพโดยใช้วิธีการของ sub texture mapping

การพัฒนาและทดสอบใช้ร่วมกับอุปกรณ์จำลองแรงสัมผัส PHANToM รวมถึง GHOST SDK ในการเชื่อมต่อระหว่างอุปกรณ์กับผู้พัฒนา และใช้ Microsoft Visual C++ 6.0 พัฒนาบนระบบปฏิบัติการ Microsoft Windows NT 4.0

Title	Development of a force-reflective simulation program for wax crayon drawing using a haptic force feedback device
Student	Mr.Teeradet Othaganont
Advisor	Assoc.Prof.Dr. Nopporn Chotikakamthorn
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2004

ABSTRACT

The development of virtual environment using haptic force feedback device is compose two parts. The part one is a development of graphic rendering module. The graphic rendering module is used to render a virtual paper and the result when a user drew. The part two is a development of force reflective module. The force reflective module is composing of two components. The component one is a friction force and component two is a force of paper texturing. Implementation of paper texture is use the bump-mapping method. The friction force is opposite the direction of motion of the virtual wax crayon. The coefficient of paper friction is 0.3 and coefficient of wax is 4.0. The texture-mapping method is use for the rendering of a virtual paper. The sub-texture mapping method is use for the drawing picture. The development program is focus on a force reflective.

In this project, the haptic force feedback device (PHANToM) has been used for development and test on Microsoft Windows NT 4.0 and GHOST API to interface with Microsoft Visual C++ 6.0.

กิตติกรรมประกาศ

ผู้เขียนขอขอบพระคุณเป็นอย่างยิ่งสำหรับ รศ.ดร.นพพร โชติกกำธร ที่ได้ให้คำแนะนำในการดำเนินโครงงาน ขอบคุณคณาจารย์ทุกท่านที่ได้สั่งสอนวิชาความรู้มาจนถึงวันนี้ และขอขอบพระคุณ คุณพ่อ คุณแม่ และพี่สาว สำหรับการสนับสนุนในทุก ๆ ด้าน

การดำเนินโครงการนี้ดำเนินการที่ห้องวิจัยสื่อประสมและระบบเสมือน ขอขอบพระคุณพี่ทุกท่าน รวมถึงน้อง ๆ ที่อยู่ในห้องวิจัยทุกท่าน ที่ได้ให้คำปรึกษา ทำให้การดำเนินโครงการนี้สำเร็จลุล่วงไปได้ด้วยดี

อนึ่งเอกสารชุดนี้จะไม่เกิดประโยชน์ใดๆ เลยถ้าไม่มีผู้สนใจทำการศึกษาและอ่านเพื่อหาความรู้ในการพัฒนาโครงการเพิ่มเติม หวังเป็นอย่างยิ่งว่าจะเป็นประโยชน์กับผู้อ่านไม่มากก็น้อย

ธีรเดช โอทกานนท์

สารบัญ

สารบัญ(ต่อ)

หน้า

บทที่

สารบัญตาราง

หน้า

ตารางที่

สารบัญรูปภาพ

หน้า

รูปที่

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่

บทที่ 1
บทนำ

1.1 โครงการพัฒนาระบบงาน

การวาดภาพด้วยสีเทียนโดยปกติแล้วจะประกอบไปด้วยกระดาษ และสีเทียน ซึ่งการฝึกฝนในการวาดภาพนั้นต้องใช้สีเทียนและกระดาษเป็นจำนวนมาก ถ้าต้องการฝึกฝนมากต้องสิ้นเปลืองอุปกรณ์มากขึ้นไปด้วย ต่อมาได้มีการพัฒนาการวาดภาพโดยใช้เครื่องคอมพิวเตอร์ ผ่านอุปกรณ์ต่างๆ ซึ่งได้สร้างความสะดวกมากยิ่งขึ้น แต่ยังติดขัดในเรื่องของความสมจริงในการวาดภาพ นั่นคือการขาดแรงสัมผัสหรือความรู้สึกถึงการสัมผัสกับแท่งสี หรือกระดาษ

จึงได้มีการพัฒนาการโปรแกรมประยุกต์ในการจำลองการวาดภาพด้วยสีเทียนด้วยอุปกรณ์จำลองแรงสัมผัส ซึ่งอุปกรณ์นี้จะสามารถให้ผู้ใช้สามารถที่จะรู้สึกถึงแรงสัมผัสต่อแท่งสีเทียน และการสัมผัสกับพื้นผิวของกระดาษ รวมถึงแรงต้านในการวาดภาพซึ่งเกิดจากสัมประสิทธิ์แรงเสียดทานของกระดาษ และของสีเทียน ในการพัฒนาโปรแกรมประยุกต์ขึ้นมาประกอบในโครงการนี้เน้นในการจำลองแรงสัมผัสในการวาดภาพ และส่วนการแสดงรูปภาพลายเส้นภายหลังจากการวาดภาพใช้เพื่อแสดงการวาดภาพไม่ได้สอดคล้องกับลักษณะของพื้นผิว ซึ่งในการพัฒนาโปรแกรมนั้นจะประกอบไปด้วย 2 ส่วนหลัก กล่าวคือส่วนของกราฟิก ใช้ในการแสดงรูปภาพของกระดาษและลายเส้นของการวาดภาพภายหลังจากการวาดภาพ และอีกส่วนหนึ่งคือการแสดงผลแรงสัมผัส ซึ่งแรงสัมผัสที่จำลองขึ้นมาจะทำให้ผู้ใช้รู้สึกถึงความขรุขระของพื้นผิวและแรงต้านในการวาดภาพซึ่งมีด้วยกัน 2 แรงคือ แรงต้านการเคลื่อนที่เป็นแรงที่จำลองการต้านการเคลื่อนที่ โดยจำลองความหนืดของกระดาษและความหนืดของสีเทียนภายหลังจากที่ได้วาดภาพแล้ว ส่วนอีกแรงหนึ่งเป็นแรงที่จำลองลักษณะความขรุขระของพื้นผิว โดยจะทำการอ่านค่าความสว่างของรูปภาพของกระดาษจำลอง มาจำลองเป็นลักษณะของความขรุขระของพื้นผิวของกระดาษ

ผลที่ได้หลังจากที่พัฒนาคือการที่มีระบบในการจำลองการวาดภาพโดยใช้สีเทียนโดยใช้อุปกรณ์จำลองแรงสัมผัส ซึ่งสามารถที่จะฝึกฝนการวาดภาพ โดยมีความสมจริงมากกว่าการวาดภาพในสภาพแวดล้อมเสมือนโดยทั่วไป และเป็นการใช้เครื่องมือจำลองการสัมผัสให้แพร่หลายมากยิ่งขึ้น

1.2 แนวทางการพัฒนา

ในการพัฒนาโครงการนี้จำเป็นต้องมีขั้นตอนในการพัฒนา ซึ่งประกอบไปด้วย

1. ศึกษาความเป็นไปได้ของการทำงานทั้งหมด
2. วิเคราะห์การทำงานของโครงการ
3. ออกแบบการทำงานของทั้งโครงการ
4. พัฒนาโปรแกรม
5. ทดสอบการทำงานของทั้งโครงการ
6. บันทึกผลการดำเนินโครงการ
7. สรุปผลการดำเนินโครงการ
8. เขียนรายงานประกอบการทำโครงงาน

1.3 วัตถุประสงค์

1. เพื่อใช้ในการจำลองการวาดภาพด้วยสีเทียนโดยใช้อุปกรณ์จำลองการสัมผัส
2. พัฒนาการจำลองแรงสัมผัสของพื้นผิวกระดาษและการวาดภาพให้มีความเสมือนจริง
3. มีเครื่องมือในการฝึกฝนการวาดภาพด้วยสีเทียน
4. สามารถนำเอาอุปกรณ์จำลองการสัมผัสมาใช้ได้หลากหลายมากยิ่งขึ้น
5. เป็นทางเลือกอีกทางหนึ่งในการฝึกฝนการวาดภาพ
6. เพื่อศึกษาและพัฒนาการทำงานโดยใช้อุปกรณ์จำลองการสัมผัส

1.4 ขอบเขตของโครงการ

ในการพัฒนาโครงการมีขอบเขตในการทำงานดังนี้

1. พัฒนาโดยใช้ Microsoft Windows NT
2. จำเป็นต้องใช้อุปกรณ์ร่วม ในการทำงานจำพวก Force Feedback Device คือ PHANToM Premium 1.0 with Encoder Finger stylus

1.5 สภาพแวดล้อมในการดำเนินโครงการ

สภาพแวดล้อมหมายรวมถึงอุปกรณ์ ระบบปฏิบัติการ ที่ใช้ในการพัฒนา และการดำเนินโครงการ ซึ่งประกอบไปด้วย

1. Microsoft Windows NT 4.0

2. Microsoft Visual C++ 6.0
3. อุปกรณ์ที่ให้ความรู้สึกได้โดยแรงสัมผัส PHANToM Premium 1.0 with Encoder Finger stylus
4. GHOST SDK Version 4.0 สำหรับติดต่อกับอุปกรณ์ PHANToM
5. OpenGL สำหรับแสดงภาพกราฟิกสามมิติ

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. มีโปรแกรมการวาดภาพด้วยสีเทียนโดยใช้อุปกรณ์จำลองการสัมผัส
2. จำลองแรงสัมผัสของพื้นผิวกระดาษและการวาดภาพให้มีความเสมือนจริง
3. มีอุปกรณ์หรือระบบเพื่อใช้เป็นทางเลือกในการฝึกฝนการวาดภาพด้วยสีเทียน
4. มีความรู้ความเข้าใจในอุปกรณ์จำลองการสัมผัสเพิ่มมากขึ้น ซึ่งสามารถนำไปประยุกต์ใช้ในงานอื่นๆได้
5. เป็นแนวทางในการพัฒนาโปรแกรมประยุกต์อื่นๆ เกี่ยวกับอุปกรณ์จำลองการสัมผัส
6. ทำให้เครื่องมือจำลองการสัมผัสใช้อย่างแพร่หลายมากยิ่งขึ้น

บทที่ 2
การพัฒนาโปรแกรมจำลองแรงสัมผัส

ทฤษฎีที่ใช้ในการดำเนินโครงการนั้น จะประกอบไปด้วย คุณลักษณะของอุปกรณ์ที่ให้ความรู้สึกได้โดยแรงสัมผัสซึ่งนำมาใช้ในโครงการ ไลบารี่ของ GHOST SDK ซึ่งนำมาติดต่อกับอุปกรณ์จำลองการสัมผัสรวมถึงควบคุมการทำงาน และทฤษฎีอีกส่วนหนึ่งนั้นจะกล่าวถึงหลักการของแบบจำลองทางกายภาพของดินเหนียว รวมถึงการจำลองของพื้นผิว และการเปลี่ยนรูปร่างของดินเหนียวเมื่อถูกกระทำโดยแรงจากผู้ใช้

2.1 อุปกรณ์ที่จำลองแรงสัมผัส

อุปกรณ์ที่ใช้ในการดำเนินโครงการคือ Personal Haptic Interface Mechanism (PHANToM) ซึ่งเป็นอุปกรณ์ที่ต้องติดตั้งเป็นโต๊ะ ออกแบบมาเพื่อจำลองการสร้างแรงต้านให้กับผู้ใช้ ดังแสดงในรูปที่ 2.1

รูปที่ 2.1 อุปกรณ์ Personal Haptic Interface Mechanism (PHANToM)

ส่วนที่สำคัญที่สุดก็คือส่วนเชื่อมต่อกับผู้ใช้ ซึ่งส่วนปลายจะมีปลอกสวมนิ้ว ซึ่งสามารถหมุนได้รอบทิศทาง ส่วนแขนก็สามารถขยับได้ อุปกรณ์ชนิดนี้มีค่าของการอิสระเป็น 6 การทำงานจะ

จำลองการสัมผัสกับวัตถุที่ปลอกหุ้มปลายนิ้ว ส่วนการส่งแรงให้กับสภาวะแวดล้อมเสมือนจะเกิดที่ปลอกสวมนิ้วเช่นกัน แต่จะไม่มีการหมุน ตัวกระตุ้นจะติดตั้งให้สมดุล นั่นคือว่า อุปกรณ์ชนิดนี้จะมีความสมดุลอยู่ในตัวอยู่แล้ว จะไม่มีการชดเชยแรงโน้มถ่วงในขณะที่มอเตอร์กำลังหมุน อุปกรณ์จะมีขนาด 8 × 17 × 25 เซนติเมตร และสร้างกำลังด้วยมอเตอร์กระแสตรงจำนวน 3 ตัวและตรวจถอดสัญญาณแบบแสง 1 ตัวซึ่งติดตั้งอยู่บนคานของตัวกระตุ้น ตัวส่งผ่านสัญญาณจะใช้สายเคเบิลพร้อมด้วยลูกรอก และทำให้กลไกการทำงานง่ายขึ้น โดยการยึดมอเตอร์ 2 ตัว กับสายเคเบิลเส้นเดียว PHANToM จะมีความเฉื่อยประมาณ 0.1 กิโลกรัม ความเสียดทานสถิตอยู่ที่ 0.1 นิวตัน ส่งแรงต้านได้สูงสุดอยู่ที่ 10 นิวตัน การควบคุมนั้นจะควบคุมผ่านคอมพิวเตอร์ ในการส่งสัญญาณนั้น เมื่อมีแรงซึ่งกระทำโดยผู้ใช้ ระบบจะตอบสนองต่อเหตุการณ์ที่ผู้ใช้กระทำนั้น ๆ โดยการสร้างแรงต้านกลับมายังผู้ใช้

2.2 GHOST SDK

General Haptics Open Software Toolkit (GHOST SDK) ของ Sensable Technology Incorporated เป็นผู้พัฒนาขึ้นมา เพื่อใช้ร่วมกับอุปกรณ์ที่ให้ความรู้สึกได้โดยแรงต้านที่ชื่อว่า PHAMToM ซึ่งพัฒนาแอปพลิเคชั่นขึ้นมาใช้อุปกรณ์จำลองแรงสัมผัส ในเอกสารนี้จะกล่าวถึง GHOST Application Programming Interface (API) ประกอบไปด้วยคลาสของ โปรแกรมภาษา C++ เชิงวัตถุ และการทำงานของ GHOST SDK ในส่วนของการพัฒนาแอปพลิเคชั่น ประกอบไปด้วย 4 ขั้นตอน คือ การสร้างฉาก สร้างแกนกลางของแอปพลิเคชั่น การเปลี่ยนข้อมูลสถานะ และสุดท้ายคือการทำความสะอาดหลังจากเสร็จสิ้นการทำงาน

ความสามารถของ GHOST SDK

1. จำลองสภาพแวดล้อมการสัมผัสโดยใช้ haptic scene graph
2. แสดงแบบจำลองลักษณะทางเรขาคณิตที่ต่างชนิดกันที่สามารถสัมผัสได้ภายใน haptic scene graph เดียวกัน
3. กำหนดคุณลักษณะของพื้นผิว เช่น แรงเสียดทาน ของแบบจำลองทางเรขาคณิต

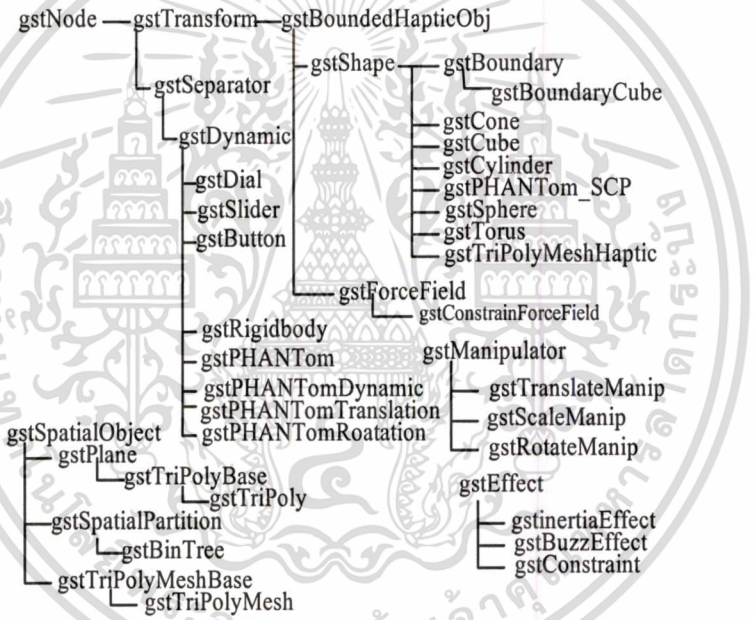
คุณสมบัติของ GHOST SDK

- สนับสนุนการเชื่อมต่อโดยการสัมผัสระหว่างมนุษย์กับคอมพิวเตอร์ รวมถึงการกระทำการสัมผัสสำหรับการปฏิสัมพันธ์ของวัตถุภายใน Haptic scene โดยใช้แรงต้านและผลการเกิดระยะช่องว่าง เช่นสปริง แรงผลักดัน การสั่น
- กลไกในการกระทำการกับเหตุการณ์ สำหรับการเข้าจังหวะสำหรับกระบวนการของการสัมผัสและกราฟิก

- สามารถใช้ VRML 2.0 ในการสร้าง Haptic scene graph
- สามารถขยายได้โดยการเพิ่มซับคลาสใหม่ ผู้พัฒนาสามารถที่จะขยาย ปรับปรุง หรือแทนที่คลาสของวัตถุได้ทั้งหมด

2.2.1 คลาสของ GHOST API

GHOST API นั้นถูกเขียนขึ้นด้วยโปรแกรมภาษา C++ เชิงวัตถุ ประกอบไปด้วยชุดของคลาสภาษา C++ การจัดการนั้นจะมีการจัดหมู่ออกเป็น 9 คลาสหลักรวมถึงชนิดของข้อมูล แสดงดังรูปที่ 2.2



รูปที่ 2.2 คลาสของ GHOST SDK

2.2.1.1 Abstract Node Classes

เป็นพื้นฐานทั้งหมดของ Haptic scene graph แต่ไม่ได้เป็นส่วนหนึ่งของ haptic scene graph ประกอบไปด้วย

ตารางที่ 2.1 องค์ประกอบของ Abstract Node Classes

Class	Description
gstNode	สำหรับโหนดทั้งหมดของ scene graph
gstTransform	เพิ่มการเปลี่ยนรูป 3มิติ และเรียกใช้โหนดอีกครั้ง
gstBoundedHapticObject	สำหรับวัตถุซึ่งสนับสนุนขอบเขตที่แน่นอน
gstShape	สำหรับโหนดการสัมผัสรูปทรงเรขาคณิต
gstdynamic	เพิ่มคุณลักษณะไปยังโหนดลูก

2.2.1.2 Hierarchical Node Classes

เป็นคลาสในการจัดกลุ่มของโหนดภายใน Haptic scene graph ประกอบไปด้วย

ตารางที่ 2.2 องค์ประกอบของ Hierarchical Node Classes

Class	Description
gstSeparator	ยอมให้มีการจัดกลุ่มของโหนดภายใต้ตัวเอง โดยให้เป็นโหนดลูก

2.2.1.3 Geometry Node Classes

เป็นคลาสที่ใช้สำหรับแบบจำลองทางเรขาคณิตชนิดต่างๆใช้ใน haptic scene graph เดียวกันได้ มีอยู่ 2 ลักษณะหลัก คือ

1. รูปทรงเรขาคณิตแบบดั้งเดิม เช่น ทรงกลม ลูกบาศก์ กรวย ทรงกระบอก
2. วัตถุที่ถูกสร้างโดยใช้โพลีกอน

รูปทรงเรขาคณิตทั้งหมดอยู่ภายใต้คลาสของ gstShape

ตารางที่ 2.3 องค์ประกอบของ Geometry Node Classes

Class	Description
gstCube	คลาสของรูปลูกบาศก์
gstcone	คลาสของทรงกรวย
gstCylinder	คลาสของทรงกระบอก
gstSphere	คลาสของทรงกลม
gstTorus	คลาสของทอรัส
gstTriPolyMeshHaptic	คลาสของการสัมผัสร่างแห 3 เหลี่ยม

2.2.1.4 Dynamic Property Node Classes

ใช้ใน Haptic scene graph เพื่อเพิ่มพฤติกรรมของวัตถุทรงเรขาคณิตในการเคลื่อนที่หรือการกำหนดตำแหน่งใหม่ในการตอบสนองต่อแรงต้าน

ตารางที่ 2.4 องค์ประกอบของ Dynamic Property Node Classes

Class	Description
gstButton	คลาสของปุ่มแบบไดนามิค
gstDial	คลาสของการหมุนแบบไดนามิค
gstSlider	คลาสของสไลเดอแบบไดนามิค
gstRigidbody	คลาสของริจิตบอร์ดี้แบบไดนามิค

2.2.1.5 Haptic interface Device Phantom Node Classes

ใช้ในการเข้าถึงข้อมูลของอุปกรณ์ในการคำนวณสถานะและตำแหน่ง หรือข้อมูลเกี่ยวกับช่องว่าง รวมถึงจุดพื้นผิวการสัมผัสของอุปกรณ์ ซึ่งสร้างมาเฉพาะอุปกรณ์ คือสำหรับ PHANToM

ตารางที่ 2.5 องค์ประกอบของ Haptic interface Device Phantom Node Classes

Class	Description
gstPHANTOM	แสดงถึงการเชื่อมต่ออุปกรณ์ Phantom ใน scene graph
GstPHANTOM_SCP	แสดงจุดพื้นผิวการสัมผัสใน scene graph โหนดนี้จะขึ้นอยู่กับโหนด gstPHANTOM
gstBoundary	เป็นคลาสพื้นฐานสำหรับขอบเขตของ Phantom
gstBoundaryCube	เป็นซับคลาสของ gstBoundary สำหรับการสร้างทรงลูกบาศก์เหมือนกับการกำหนดขนาด

2.2.1.6 Effect Node Classes

ใช้ในการสร้างความรู้สึกในการสัมผัสกับช่องว่างของอากาศ สำหรับการพัฒนาการเชื่อมต่อระหว่างผู้ใช้กับคอมพิวเตอร์ ในการนี้ใช้ในการกระตุ้นการสัมผัสไม่ใช่การกระทำโดยตรงกับวัตถุ

ตารางที่ 2.6 องค์ประกอบของ Effect Node Classes

Class	Description
gstEffect	เป็นคลาสหลักสำหรับอุปกรณ์ Phantom เมื่อเกิดเหตุการณ์ของช่องว่างของอากาศ
gstBuzzEffect	สร้างรายงานเช่นการสั่นสำหรับอุปกรณ์ Phantom เมื่อถึงจุดสิ้นสุด
gstConstrainEffect	กำหนดจุดสิ้นสุดของอุปกรณ์ เป็นในรูปของจุด เส้น หรือ ระนาบ โดยการเชื่อมต่อเข้ากับสปริง
gstInertiaEffect	เพิ่มความเฉื่อยและหรือ เพิ่มความหนืด สำหรับบอกจุดสิ้นสุดของอุปกรณ์

2.2.1.7 Manipulator Node Classes

ใช้ในการเปลี่ยนแปลงรูปร่างของวัตถุ เช่น translate rotate หรือ scale

ตารางที่ 2.7 องค์ประกอบของ Manipulator Node Classes

Class	Description
gstManiplator	เป็นคลาสหลักสำหรับการกระทำกับโหนด
gstTranslateManip	กระทำการทำงานในโหมด translate
gstRotateManip	กระทำการทำงานในโหมด Rotate
gstScaleManip	กระทำการทำงานในโหมด Scale

2.2.1.8 The Phantom Dynamic Node Classes

ใช้กับอุปกรณ์ Phantom ในการควบคุมตำแหน่งและการกำหนดตำแหน่งของซับทรีของโหนดนี้

ตารางที่ 2.8 องค์ประกอบของ The Phantom Dynamic Node Classes

Class	Description
gstPHANTomDynamic	เป็นคลาสหลักสำหรับคลาส Phantom Dynamic
gstTranslateDynamic	ใช้อุปกรณ์ Phantom ในการควบคุมตำแหน่งของซับทรีใน scene graph
gstRotateDynamic	ใช้ในการควบคุมตำแหน่งและการเปลี่ยนแปลงตำแหน่งของซับทรีใน scene graph

2.2.1.9 Scene Classes

ใช้ในกระบวนการพื้นฐานของ GHOST SDK การแสดงการจำลองการทำงานวนไปเรื่อยๆ Haptic scene graph สามารถใช้ในการปฏิสัมพันธ์เมื่อมีการกระทบกันของวัตถุ หรือเมื่อเรียก startServoLoop()

ตารางที่ 2.9 องค์ประกอบของ Scene Classes

Class	Description
gstScene	จัดการกับ haptic scene graph และการจำลองแบบจำลอง
gstDeviceIO	เข้าถึงอุปกรณ์ Phantom ในระดับต่ำ

2.2.1.10 Data Type

ชนิดของข้อมูลหลายๆชนิดที่ใช้ใน GHOST API

ตารางที่ 2.10 ชนิดของข้อมูลที่ใช้ใน GHOST API

Class	Description
gstNodeName	คาสเกี่ยวกับตัวอักษรในการจัดเก็บชื่อสำหรับโหนด
gstPlane	คลาสเกี่ยวกับระนาบ
gstPoint	คลาสเกี่ยวกับคาร์ทีเซียน 3 มิติ
gstTransformMatrix	โฮโมจีเนียส 4*4 ทรานสฟอร์เมชั่นเมตริกซ์
gstVertex	คลาสของร่างแหจุดยอด 3 จุด
gstEdge	คลาสของร่างแห 3 ด้าน
gstTriPoly	คลาสของร่างแห 3 โพลีกอน
gstTriPolyMesh	คลาสของร่างแห 3ร่างแห
gstRay	คลาสเส้นรัศมี
gstLine	คลาสของเส้น
gstVector	คลาสของคาร์ทีเซียนเวกเตอร์ 3มิติ

2.2.2 การใช้งาน GHOST SDK

โครงสร้างของการทำงานของ GHOST SDK นั้นประกอบด้วย 2 ส่วน ส่วนแรกคือใน ส่วนของกระบวนการสร้างแอปพลิเคชั่น (Application Process) และอีกส่วนหนึ่งคือในส่วนของกระบวนการการสัมผัส (Haptic Process) ดังรูปที่ 2.3

1. **กระบวนการสร้างแอปพลิเคชั่น**

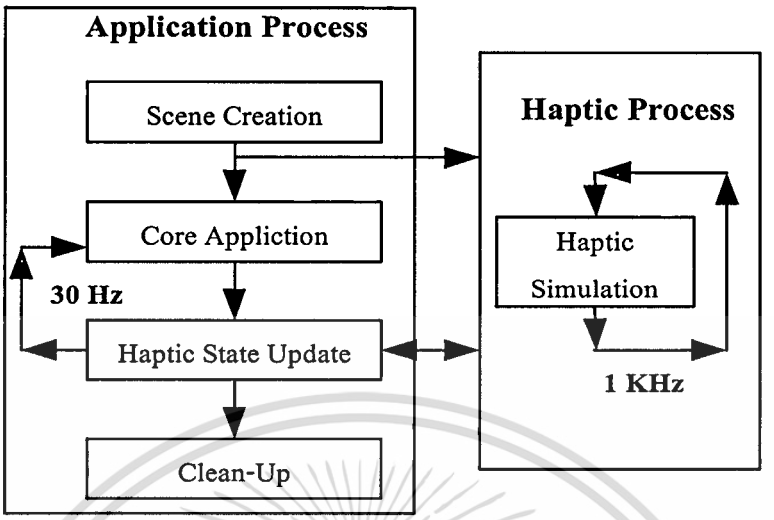
กระบวนการสร้างแอปพลิเคชั่น หรือ แอปพลิเคชั่นลูป ทำหน้าที่หลัก คือ

- แสดงกราฟิกเชื่อมต่อกับผู้ใช้ (Graphic User Interface)
- สร้างหรือจัดการ haptic scene graph เพื่อสร้างสภาพแวดล้อมการสัมผัส
- เริ่ม หยุด หรือ จัดการ กระบวนการจำลองการสัมผัสผ่านทาง Scene Node
- แสดงกราฟิกจากข้อมูลที่ได้รับผ่านทางกระบวนการการสัมผัส

2. **กระบวนการการสัมผัส**

กระบวนการการสัมผัสหรือ Servo Loop ทำหน้าที่ การจำลองสภาพแวดล้อมการสัมผัส ซึ่งถูกกำหนดโดย Haptic Scene Graph และ Scene Node ที่ได้สร้างและกำหนดในส่วนของกระบวนการสร้างแอปพลิเคชั่น โดยการจำลองแสดงการทำงานนั้น จะเป็นการวนการทำงานซ้ำๆ ในการคำนวณเพื่อกำหนดแรงในการตอบสนองผ่านอุปกรณ์ PHANToM ซึ่งภายในการทำซ้ำๆ นั้นจะมีการทำงาน ดังนี้

- ปรับปรุงสถานะของ Phantom Node ใน Haptic Scene Graph
- ตรวจสอบการชนกันของ Phantom Node กับ Geometry Nodes
- คำนวณแรงที่ให้อุปกรณ์แสดงต่อ Phantom Node
- ถ้าในกระบวนการสร้างแอปพลิเคชั่นใดมีการกำหนด Graphic Callback หรือ Event Callback สำหรับโหนดใดๆ จะมีการเตรียมข้อมูลสำหรับการ Callback สำหรับโหนดนั้น เมื่อโหนดนั้นมีสถานะใหม่



รูปที่ 2.3 การสร้างแอปพลิเคชั่นโดยใช้ GHOST SDK

2.2.3 โครงสร้างการโปรแกรม

การพัฒนาแอปพลิเคชั่นโดยใช้ GHOST SDK มีโครงสร้างการโปรแกรม ดังนี้

1. สร้างแอปพลิเคชั่นตามที่ต้องการ
2. สร้าง Haptic Scene Graphs และกำหนด Graphic Callbacks หรือ Event Callbacks
3. สร้าง Scene Node และกำหนด Haptic Scene Graphs ที่ใช้
4. ควบคุมกระบวนการการสัมผัสผ่าน Scene Node
5. เมื่อต้องการข้อมูลของกระบวนการการสัมผัส ให้เรียกผ่าน Scene Node เพื่อให้กระบวนการการสัมผัสเตรียมข้อมูลก่อน และจะเรียกกลับผ่าน Callback ที่กำหนดใน Haptic Scene Graphs
6. เมื่อเลิกใช้ ทำการยกเลิกการจองทรัพยากรต่างๆ

2.2.4 การควบคุมการจำลอง (simulations)

ในการควบคุมการจำลองของกระบวนการสัมผัส นั้นจะควบคุมผ่าน Scene Node ซึ่งสร้างจาก gstScene Class ทำหน้าที่ในการกำหนด Haptic Scene Graph ที่จะทำการจำลอง โดยสั่งเริ่มหรือหยุด หรือสั่งปรับปรุงข้อมูลกระบวนการสัมผัส เมื่อกระบวนการของแอปพลิเคชั่นต้องการใช้ในการทำงานนั้นจะมีเพียงโหนดเดียวเท่านั้นในแต่ละแอปพลิเคชั่น

```
gstScene * scene = new gstScene;
scene->setRoot(root);
scene->startServoloop();
        ....
scene->updateGraphics();
scene->updateEvents();
        ...
scene->stopServoLoop();
```

รูปที่ 2.4 ตัวอย่างการโปรแกรมการควบคุมการจำลองการทำงาน

จากรูปเป็นการสร้าง Scene Node โดยกำหนด root ให้เป็น Haptic Scene Graph ที่จะทำการจำลองการทำงาน เริ่มด้วยการสั่งเริ่มการทำงานของกระบวนการการสัมผัส ต่อด้วยการปรับปรุงข้อมูลของการสัมผัส ใช้สำหรับการ Callback ที่ได้ลงทะเบียนไว้ และสุดท้ายเป็นการสั่งหยุดการจำลอง

2.3 Open Graphic Library (OpenGL)

OpenGL เป็นซอฟต์แวร์ที่ใช้ในการติดต่อกับอุปกรณ์ทางด้านการแสดงผลรูปภาพทั้งในส่วนของจอภาพและการ์ดแสดงผล จะประกอบไปด้วยโพรซีเจอร์และฟังก์ชัน เพื่อใช้ในการโปรแกรมเพื่อสร้างกราฟิกที่มีคุณภาพสูง โดยเฉพาะกับการสร้างกราฟิกแบบ 3 มิติ ซึ่ง OpenGL นั้นเริ่มแนะนำในปี พ.ศ.2534 โดยบริษัท Siligon Graphics Incorporated (SGI)

2.3.1 คุณลักษณะเด่นของ OpenGL

1. เป็นมาตรฐานในด้านกราฟิก กล่าวคือ รายละเอียดของ OpenGL ถูกกำหนดและจัดการโดยสมาคมอิสระ Architecture Review Board (ARB)

2. มีเสถียรภาพ เนื่องจากมีการพัฒนากันมาอย่างยาวนาน รวมถึงได้มีการทดสอบและเป็นที่ยอมรับของนักพัฒนากราฟิกอย่างแพร่หลาย

3. มีความน่าเชื่อถือและไม่ขึ้นกับแพลตฟอร์ม OpenGL นั้นไม่คำนึงถึงระบบปฏิบัติการแอปพลิเคชั่นที่ใช้ OpenGL สามารถทำงานได้ตั้งแต่คอมพิวเตอร์ขนาดเล็ก เวิร์คสเตชั่น จนถึงซุปเปอร์คอมพิวเตอร์

4. ความง่ายในการใช้งาน OpenGL มีคำสั่งต่างๆมากมายซึ่งได้ถูกจัดเป็นโครงสร้างอย่างดี

ทำให้มีความง่ายในการใช้งาน

2.3.2 สถาปัตยกรรมของ OpenGL

OpenGL ถูกออกแบบในลักษณะของสถานะ (state) ที่คอยควบคุมการแสดงผลภาพ กล่าวคือ เมื่อกำหนดรูปทรงที่จะวาดแล้วผลลัพธ์ที่ได้นั้นขึ้นอยู่กับการกำหนดสถานะก่อนหน้านั้นอย่างไร เช่นการกำหนดสี รูปแบบ รูปทรง แสงเงา เมื่อรวมกันเข้าจะเกิดการแสดงผลที่สวยงามขึ้น

2.3.3 การตั้งชื่อของฟังก์ชันของ OpenGL

OpenGL นั้นมีแบบแผนในการตั้งชื่อเพื่อให้ผู้พัฒนาสามารถทราบได้ว่ามาจากฟังก์ชันไหน ไลบารี่อะไร รวมถึงจำนวนอาร์กิวเมนต์เป็นชนิดอะไร ซึ่งรูปแบบโครงสร้างมีดังนี้

<Library Type><Root command><Number of arguments><Data Type>

Library Type	บอกว่าใช้ไลบารี่ไหน
Root Command	เป็นรากของคำสั่ง
Number of arguments	จำนวนของพารามิเตอร์
Data Type	ชนิดของตัวแปร

รูปที่ 2.5 รูปแบบโครงสร้างการตั้งชื่อของ OpenGL

จากรูปที่ 2.5 ตัวอย่างในการตั้งชื่อ เช่น glVertex3f รากของคำสั่งคือ Vertex ส่วนขึ้นต้นด้วย gl คือมาจากไลบารี่ของ gl ส่วน 3f หมายความว่ามีอาร์กิวเมนต์ 3 ตัวเป็นชนิด float

2.3.4 ชนิดข้อมูลของ OpenGL

OpenGL ใช้ชนิดของข้อมูล ซึ่งกำหนดขึ้นเอง จะขึ้นด้วย GL ตามด้วยชนิดของข้อมูล โดยมีรายละเอียดดังนี้

ตารางที่ 2.11 ชนิดของข้อมูลที่ใช้ใน OpenGL

Suffix	Data Type	Typical C type	OpenGL Type Name
b	8-bit integer	Signed char	GLbuyte
s	16-bit integer	Short	GLshort
i	32-bit integer	Int or long	GLint ,GLsizel
f	32-bit floating point	Float	GLfloat , GLclampf
d	64-bit floating point	Double	GLdouble , GLclampd
ub	8-bit unsigned number	Insigned char	GLubyte , GLboolean
us	16-bit unsigned number	Unsigned short	GLushort
ui	32-bit unsigned number	Unsigned int or unsigned long	GLuint ,Glenum , GLbitfield

2.3.5 รูปทรงพื้นฐานของ OpenGL

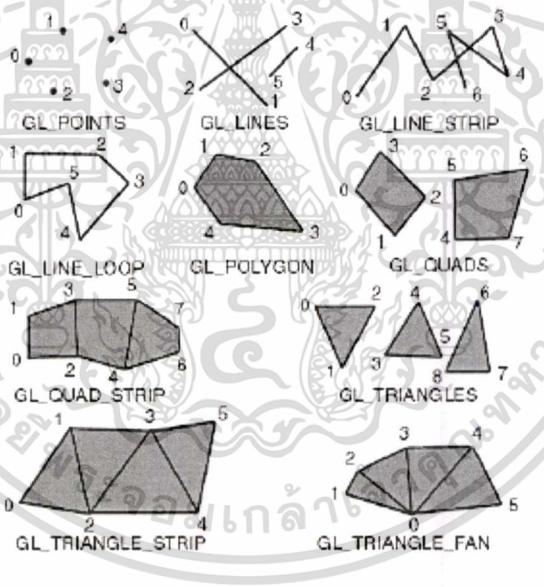
รูปทรงพื้นฐานของ OpenGL นั้น สร้างได้จากการกำหนดจุดยอด และลำดับการเรียงกันของจุดยอด ซึ่งกำหนดโดยฟังก์ชัน glVertex3f ( ) การกำหนดเซตของจุดยอด เพื่อกำหนดเป็นรูปทรงต่างๆ นั้น โดยจะเริ่มจากการกำหนดประเภทของรูปทรงที่ต้องการ ซึ่งเป็นส่วนหนึ่งของพารามิเตอร์ glBegin () ตามด้วยชุดของจุดยอด และสุดท้ายคือ glEnd ()

ตัวอย่างการสร้างรูปทรงนั้น จะแสดงการสร้างรูปทรงสามเหลี่ยมซึ่งจะมีจุดยอดทั้งหมด 3 จุด ซึ่งต้องใช้พารามิเตอร์ 3 ตัวเป็นประเภท float รวมทั้งสิ้น 3 เพื่อสร้างรูปสามเหลี่ยม แสดงได้ดังนี้

```
Begin( GL_TRIANGLES);
        Vertex3f(1.0,1.0,1.0);
        Vertex3f(1.0,-1.0,1.0);
        Vertex3f(1.0,1.0,-1.0);
End();
```

รูปที่ 2.6 ตัวอย่างการสร้างรูปสามเหลี่ยมใน OpenGL

ต่อไปจะเป็นการแสดงรูปทรงพื้นฐานต่างๆ ที่ใช้ใน OpenGL ดังแสดงในรูปที่ 2.7



รูปที่ 2.7 รูปทรงพื้นฐานที่ใช้ใน OpenGL

2.3.6 การกำหนดสถานะ

การกำหนดสถานะนั้นใช้เพื่อในการเรียกใช้งานส่วนเพิ่มเติมต่างๆ โดยจะมีการยอมให้แสดงผลหรือไม่ยอมให้แสดงผล ฟังก์ชันที่ใช้ในการยอมให้มีการแสดงผลคือ glEnable () ส่วนฟังก์ชันที่ไม่ยอมให้มีการแสดงผลนั้นคือ glDisable() ตัวอย่างเช่น glEnable(GL_LIGHTING) หมายถึง ยอมให้มีการแสดงผลของการใช้แสง

ส่วนการกำหนดสถานะนั้นจะเรียกใช้ตามฟังก์ชันนั้น เช่นการใส่สีให้กับวัตถุ จะใช้ glColor3f(...)

2.4 แรงต้านในการวาดภาพ

แรงต้านที่จะดำเนินการนั้นประกอบไปด้วย 2 แรง คือแรงต้านการเคลื่อนที่ ซึ่งจะแสดงถึงแรงต้านในการในการเคลื่อนที่ เป็นการจำลองแรงต้านการเคลื่อนที่เพื่อสร้างแรงหนืดหรือแรงต้านการเคลื่อนที่ของการวาดภาพ เมื่อทำการวาดภาพกับกระดาษเปล่าสัมประสิทธิ์แรงเสียดทานจะเป็นค่าหนึ่ง และต่อเมื่อทำการวาดภาพซ้ำ ในจุดเดิมที่ได้ทำการวาดภาพแล้วจะทำการจำลองสัมประสิทธิ์แรงเสียดทานอีกค่าหนึ่งเพื่อให้ผู้ใช้รู้สึกความหนืดของขี้ผึ้ง ส่วนอีกแรงหนึ่งนั้นเป็นแรงซึ่งจำลองลักษณะของกระดาษ เพื่อทำการจำลองลักษณะพื้นผิวของกระดาษจำลอง

2.4.1 การจำลองพื้นผิว

ในเทคนิคการจำลองพื้นผิวสามารถทำได้โดยการเพิ่มความขรุขระให้กับพื้นผิวที่เรียบ ซึ่งเหมือนกับการจำลองพื้นผิวในคอมพิวเตอร์กราฟิก ผลสุดท้ายของการจำลองพื้นผิวคือการตอบสนองทิศทางของแรงให้กับผู้ใช้

2.4.1.1 ขนาดของแรง

เมื่อไม่มีความขรุขระบนพื้นผิวปกติ จะใช้กฎของความยืดหยุ่นในการตอบสนองต่อแรงซึ่งจะขึ้นอยู่กับความลึกที่ทะลุผ่านเข้าไปในวัตถุ เมื่อลักษณะมีความสูงเพิ่มขึ้น ทำให้ลักษณะของการจำลองของแรงก็เปลี่ยนแปลงไป

2.4.1.2 ทิศทางของแรงต้าน

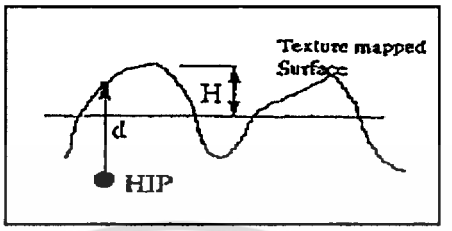
ในการออกแบบทิศทางของแรงต้านในการจำลองแรงสัมผัส ดัดแปลงมาจาก bump-mapping ซึ่งใช้ในคอมพิวเตอร์กราฟิก ซึ่งเป็นการสร้างลักษณะของพื้นผิวให้มีความสูงโดยการทำให้ลักษณะของพื้นผิวปกติเปลี่ยนแปลงไป ขั้นแรกนั้นต้องดูถึงลักษณะของพื้นผิวปกติ ซึ่งจะทำให้เกิดลักษณะของแรง ทำได้โดยยึดหลักของพื้นผิวเริ่มต้นและการเปลี่ยนแปลงของความสูงนั้นจุดนั้นความสูงนั้นทำให้พื้นผิวมีความสูงต่ำ ดังสมการ

$$\overrightarrow{M} = \overrightarrow{N_s} - \nabla h + (\nabla h.\overrightarrow{N_s})\overrightarrow{N_s} \qquad (1)$$

เมื่อ

$\overrightarrow{M}$	คือ	พื้นผิวภายหลังจากการที่ทำ texture mapping
$\overrightarrow{N_s}$	คือ	surface normal ก่อนที่จะทำการ map texture
∇h	คือ	อัตราการเปลี่ยนแปลงของพื้นผิว

เมื่อได้ลักษณะของพื้นผิวแล้วนั้นต่อไปจะทำการคำนวณหาแรงที่จะทำการตอบสนองให้กับผู้ใช้งานดังสมการ



รูปที่ 2.7 ลักษณะของพื้นผิวภายหลังจากที่ทำการแมปพื้นผิว

$$\vec{F} = \mathrm{d}\ \vec{M} \qquad (2)$$

เมื่อ

$\vec{F}$	คือ	แรงที่จะตอบสนองให้กับผู้ใช้
$\vec{M}$	คือ	พื้นผิวภายหลังจากการที่ทำ texture mapping
d	คือ	ความลึกของการทะลุเข้าไปในพื้นผิว

2.4.2 แรงต้านการเคลื่อนที่

แรงต้านจะดำเนินการโดยการคำนวณจากจำนวนครั้งในการวาดภาพ ซึ่งเทียบเคียงให้สอดคล้องกับจำนวนขี้ผึ้งที่ติดลงบนกระดาษ เมื่อเริ่มแรกของการวาดภาพกระดาษซึ่งไม่มีการติดลงบนกระดาษเป็นการจำลองสัมประสิทธิ์แรงเสียดทานของกระดาษ และเมื่อทำการวาดภาพซ้ำในจุดเดิมยึดถือเอาว่าได้มีขี้ผึ้งติดอยู่ที่พื้นผิวของกระดาษแล้ว ให้ทำการจำลองแรงสัมประสิทธิ์แรงเสียดทานของขี้ผึ้งเข้ามาแทนที่

$$\vec{F_F} = \mu \vec{F_N} \qquad (3)$$

เมื่อ

$\vec{F}_F$	คือ	แรงต้านที่แสดงให้กับผู้ใช้
$\vec{F}_N$	คือ	ทิศทางปกติของพื้นผิว
μ	คือ	สัมประสิทธิ์ของแรงต้านบนกระดาษ

ด้วยการใช้ค่าความสูงของพื้นผิว จะใช้ค่าความสูงในการกำหนดความสูงของกระดาษ ในการต้านการเคลื่อนที่ของแท่งสีเทียน และการคำนวณแรงต้านการเคลื่อนที่สำหรับกระดาษ ค่าของ μ ขึ้นอยู่กับเงื่อนไขว่าแท่งสีกระทำกับกระดาษที่ว่างเปล่าหรือกับกระดาษที่มีสีอยู่แล้ว ค่า μ_{paper} มีค่าเท่ากับ 0.3 และ ค่า μ_{wax} ในแต่ละชั้นของการเขียนคือ 4.0

บทที่ 3

การออกแบบระบบ

การออกแบบระบบโครงการการพัฒนาโปรแกรมจำลองแรงสัมผัสของการวาดภาพด้วยสีเทียน โดยใช้อุปกรณ์จำลองการสัมผัสแบ่งเป็น 2 ส่วน คือระบบย่อยของการตอบสนองต่อแรงสัมผัส กับส่วนของการแสดงผล 3 มิติ ซึ่งใช้ MFC (Microsoft foundation Class) ใน Microsoft Visual C++ 6.0 ในการพัฒนาร่วมกับ GHOST (General Haptics Open Software Toolkit) SDK (Software Developer's Kit) 4.0 ส่วนการแสดงผลใช้ OpenGL ในการแสดงผล 3 มิติ ซึ่งรูปแบบการสัญลักษณ์ในการออกแบบจะใช้ CRC (Class/Responsibilities/Collaborators)

3. 1 CRC (Class/Responsibilities/Collaborators)

CRC เป็นการออกแบบคลาส หรือ อ๊อบเจ็ค ที่มีอยู่ในระบบว่าเป็นคลาส อะไรสืบทอดมาจากคลาสอะไร มีหน้าที่ทำอะไรบ้าง และการทำงานของคลาสนั้นต้องทำงานร่วมกับคลาสหรืออ๊อบเจ็คใดบ้าง ซึ่งชื่อของคลาสจะอยู่บนซ้ายของตาราง ตามด้วยเครื่องหมายตามด้วยเครื่องหมายโคลอน และชื่อคลาส ที่สืบทอดมา (ถ้ามีการสืบทอด โดยภายในตารางแยกเป็นด้านซ้ายแสดงหน้าที่ของคลาสนั้นซึ่งถูกกำกับด้วยหมายเลข ส่วนด้านขวาเป็นคลาส ที่ทำงานร่วมกันเพื่อให้หน้าที่ในข้อนั้น ๆ สำเร็จ

3.1.1 CRC ของ CCrayonApp

ตารางที่ 3.1 CRC ของ CCrayonApp

CCrayonApp: CHapticApp	
1.Create SDI Framework 2. start_program 3. end_program	

จาก CRC ของ CCrayonApp มีหน้าที่

1. สร้าง SDI Framework คือ CCrayonDoc CMainFrame CCrayonView ตามโครงสร้างของ MFC SDI Framework

2. start_program เริ่มการทำงานของโปรแกรม ซึ่งจะทำการกำหนดค่าการทำงานของโปรแกรม รวมถึงการสร้างซีนกราฟของ Haptic process

3. end_program หยุดการทำงานของโปรแกรม ทำการลบ Phantom Node ที่ซึ่งถูกสร้างขึ้นมาเพื่อใช้งาน

3.1.2 CRC ของ CCrayonDoc

ตารางที่ 3.2 CRC ของ CCrayonDoc

CCrayonApp: CHapticDoc	
1. DECLARE_MESSAGE_MAP 2. NewDocument	

จาก CRC ของ CCrayonDoc มีหน้าที่

1. ทำการจับคู่กับ Windows Message ที่ได้ประกาศไว้

2. สร้างเอกสารหรือ SDI ใหม่

3.1.3 CRC ของ CCrayonView

ตารางที่ 3.3 CRC ของ CCrayonView

CCrayonView : CHapticView	
1. EnableServoLoop 2. QueryPHANToMNames 3. QueryCursorPos 4. TermGraphics 5. ResizeGraphics 6. UpdateGraphics	

ตารางที่ 3.3 CRC ของ CCrayonView (ต่อ)

CCrayonView : CHapticView	
7. InitGraphics	
8. ProgramDone	
9. EndProgram	CCrayonApp
10. StartProgram	CCrayonApp

จาก CRC ของ CCrayonView มีหน้าที่

1. การบอกให้ทำการเริ่มต้นการวนลูปการทำงานของอุปกรณ์จำลองการสัมผัส
2. ร้องของชื่ออุปกรณ์จำลองการสัมผัสเพื่อใช้ในการทำงาน
3. ร้องของตำแหน่งของ HIP (Haptic Interface Point)
4. ยกเลิกการทำงานของกราฟิก
5. ปรับเปลี่ยนขนาดของกราฟิก
6. ปรับเปลี่ยนสถานะของการฟิก เมื่อมีการเปลี่ยนแปลง เช่นมีการวาดภาพ
7. กำหนดสถานะเริ่มต้นของกราฟิก
8. ทำลายซีนกราฟก่อนการออกจากโปรแกรม
9. ยกเลิกการทำงาน ก่อนออกจากโปรแกรม
10. เริ่มต้นการทำงานของโปรแกรม

3.1.4 CRC ของ CMainFrame

ตารางที่ 3.4 CRC ของ CMainFrame

CMainFrame : CHapticFrame	
1. DECLARE_MESSAGE_MAP	
2. Manage View Menu Toolbar and Statusbar	

จาก CRC ของ CMainFrame มีหน้าที่

1. ทำการจับคู่กับ Windows Message ที่ได้ประกาศไว้
2. จัดการกับ View Menu Toolbar and Statusbar เพื่อใช้ในการแสดงผล

3.1.5 CRC ของ CPlane

ตารางที่ 3.5 CRC ของ CPlane

CPlane : gstShape	
1. LoadGLTextures 2. DrawScene 3. DrawPlane 4. DrawTexture 5. collisionDetect	 Texture CTextureEffect

จาก CRC ของ CPlane มีหน้าที่

1. โหลดรูปของพื้นผิวของกระดาษขึ้นมาเก็บไว้ในหน่วยความจำ
2. ทำการวาดรูปในส่วนของการแสดงผลของกระดาษ และการวาดภาพ
3. แสดงผลในส่วนของการแสดงกระดาษ
4. ทำการวาดภาพของพื้นผิวที่ได้ทำการปรับปรุงซึ่งเกิดจากการวาดภาพโดยใช้บริการจากคลาส Texture
5. ตรวจสอบการชนกันของ HIP กับกระดาษที่จำลองขึ้นมาแล้วทำการจำลองแรงสัมผัสซึ่งเรียกใช้บริการจากคลาส CTextureEffect

3.1.6 CRC ของ Texture

ตารางที่ 3.6 CRC ของ Texture

CPlane : gstShape	
1. LoadTexture 2. LoadSubTexture	

จาก CRC ของ Texture มีหน้าที่

1. โหลดรูปของพื้นผิวหลักขึ้นมาแสดงผล
2. โหลดรูปของพื้นผิวรองซึ่งเป็นส่วนแสดงผลสีที่ได้วาดลงไปบนกระดาษจำลอง

3.1.7 CRC ของ CTextureEffect

ตารางที่ 3.7 CRC ของ CTextureEffect

CTextureEffect :gstEffect	
1. calcEffectForce 2. setEffect	gstPHANToM

จาก CRC ของ CTextureEffect มีหน้าที่

1. คำนวณแรงซึ่งจะตอบสนองสู่ผู้ใช้
2. กำหนดแรงให้กับอุปกรณ์เพื่อใช้ในการแสดงผล

3.2 The HapticView Framework

เป็นการพัฒนาเค้าโครงของโปรแกรมเพื่อให้การพัฒนาไม่ขึ้นกับแพลตฟอร์มใดๆ ซึ่งในเค้าโครงนี้จะประกอบไปด้วยฟังก์ชั่นในการจัดการกับ GHOST SDK เพื่อใช้ในการทำงานร่วมกันกับ Windows/MFC หรือ X Windows แต่เค้าโครงนี้ออกแบบมาเพื่อทำงานร่วมกันกับ OpenGL เท่านั้น ไม่สามารถทำงานร่วมกับกราฟิกไลบารี่อื่นๆ

ในการที่จะใช้งาน HapticView Framework นั้นจะต้องทำการ derive method มาจากคลาส HapticView ซึ่งคลาสนี้เป็นส่วนเชื่อมต่อเพื่อทำงาน ประกอบไปด้วย Interface Function ดังนี้

```
virtual void StartProgram(BOOL bPHANToMMouseEnabled);
virtual void EndProgram();
virtual BOOL ProgramDone();
virtual void InitGraphics();
virtual void UpdateGraphics();
virtual void ResizeGraphics(int cx, int cy);
virtual void TermGraphics();
virtual void EnableServoLoop(BOOL bEnable);
virtual LPCSTR* QueryPHANToMNames();
virtual BOOL QueryCursorPos(double* pX, double* pY, double* pZ);
```

3.2.1 ภาพรวมของ Framework

เมื่อเริ่มต้นสร้างตัวเค้าโครงจะทำการสร้าง rendering context ซึ่ง InitGraphic() จะทำการเรียกใช้งาน ซึ่งการทำงานนี้จะกำหนดการทำงานของ OpenGl เช่น depth buffering culling lighting ในขณะที่เค้าโครงกำลังทำงานตัวเค้าโครงจำเป็นต้องทราบถึงตำแหน่งของอุปกรณ์ซึ่งในขณะที่กำลังสร้างอยู่นั้นจะทำให้มีการกำหนดให้มีการกำหนดตำแหน่งของอุปกรณ์รวมถึงมีการกำหนดชื่อของอุปกรณ์ เช่น Default Phantom ก่อนที่จะสร้าง View เสร็จจะมีการเรียกใช้ ResizeGraphics() ซึ่งฟังก์ชั่นนี้จะกำหนดขนาดของ view ผู้พัฒนาจะต้องทำการปรับแต่ง OpenGL viewport หรืออื่นๆที่ผู้พัฒนาระบบต้องการ ภายหลังจากการสร้าง view แล้วต่อไปเป็นการเรียกใช้ StartProgram() ซึ่งจะมีการแสดงรูปภาพให้ทำการปรับตั้งตำแหน่งของ Phantom ให้อยู่ในสถานะ reset ซึ่งถ้าตั้งอยู่ในตำแหน่งแล้วให้กด Enter หลังจากนั้น StartProgram() จะถูกเรียกใช้ ซึ่งจากจุดนี้เป็นทางผ่านในการเข้าถึงการกำหนดค่าต่างๆของโปรแกรม ผู้พัฒนาต้องสร้างซีนกราฟรวมถึงกราฟิกต่างๆที่ต้องการ เมื่อกระทำการเสร็จสิ้น HapticView จะเรียกใช้ EnebleServoLoop() และทำการเรียกใช้ UpdateGraphics() เป็นระยะ ๆ ในการวาดภาพของ OpenGL

ในขณะที่กำลังทำงานอยู่นั้น ProgramDone() จะทำงานเป็นระยะเพื่อให้แน่ใจว่า servoloop ยังทำงานอยู่ ถ้าขั้นตอนนี้คืนค่าเป็น TRUE โปรแกรมจะทำการลบค่าทั้งหมดแล้วปิดโปรแกรม

ท้ายที่สุดเมื่อจะออกจากโปรแกรมจะมีการเรียกใช้งานทั้งส่วนของ TermGraphics() และ EndProgram() ซึ่งจะใช้ในการคืนหน่วยความจำให้กับระบบทั้งในส่วนของกราฟิกและHaptic

3.2.2 HapticView Interface Function

1. StartProgram(BOOL bPHANToMMouseEnabled);

Function Prototype:

virtual void StartProgram(BOOL bPHANToMMouseEnabled);

Function Description:

เป็นจุดสำคัญในการเข้าถึงการกำหนดทั้งส่วนของกราฟิกและHaptic ซึ่งจะต้องมีการสร้าง instance ของ gstScene และ gstPHANToM ส่วนของ bPHANToMMouseEnabled นั้นเพื่อให้รู้ถึงต้องทำการ reset Phantom นอกจากนี้ต้องทำการกำหนดค่าในส่วนของ gstPHANToM constructor เป็น TRUE ตัวอย่างเช่น

```
BOOL bResetPHANToM = !bPHANToMMouseEnabled;
gstPHANToM *myPHANToM = new gstPHANToM("Default PHANTOM",bResetPHANToM);
```

2. EndProgram()

Function Prototype:

virtual void EndProgram();

Function Description:

ทำการคืนค่าที่ Haptic ใช้งานให้กับระบบ ซึ่งจะมีทั้งการลบ instanceของ gstScene ซึ่งจะลบโหนดลูกทั้งหมดที่อยู่ภายใต้ root

3. ProgramDone()

Function Prototype:

virtual BOOL ProgramDone();

Function Description:

จะมีการเรียกใช้อยู่เป็นระยะๆเพื่อทำการตรวจสอบว่า servoloop ยังทำงานอยู่หรือไม่ ถ้าฟังก์ชั่นนี้คืนค่าเป็น TRUE จะทำการออกจากโปรแกรม

4. InitGraphics()

Function Prototype:

Virtual void InitGraphics();

Function Description:

จะทำการสร้าง view และ rendering context ของ OpenGL ใช้ฟังก์ชั่นนี้ในการตั้งค่าในการแสดงผลเช่น depth buffering culling lighting

5.UpdateGraphics();

Function Prototype:

virtual void UpdateGraphics();

Function Description:

HapticView เรียกใช้งานฟังก์ชั่นนี้ในการ redraw ของ view ที่ประมาณ 30 Hz ซึ่งจะคอยจับการทำงานของโพรเซสกราฟิก เช่น การเรียกใช้ upDateGraphics หรือ updateEvents ภายในซีน-กราฟ และทำการวาดวัตถุที่อยู่ในซีนนั้น HapticView จะทำการ clear ค่าของ OpenGL เช่น color หรือ depth buffers ก่อนที่จะทำการเรียก UpdateGraphics() ภายหลังจากคืนค่าแล้วจะทำการ swap buffer ให้กับระบบ

6. ResizeGraphics(int cx, int cy)

Function Prototype:

virtual void ResizeGraphics(int cx, int cy);

Function Description:

จะทำงานเมื่อเกิดเหตุการณ์การเปลี่ยนขนาดของ view ซึ่ง parameter ทั้ง 2 ตัวนั้นเป็นขนาดใหม่ในแกน x และ y ซึ่งการโปรแกรมส่วนน้อยที่สุดควรจะมีคือ

glViewport(0, 0, cx, cy);

7. TermGraphics ()

Function Prototype:

virtual void TermGraphics();

Function Description:

จะทำการคืนค่าที่ใช้ในการแสดงผลกราฟิก เช่น display lists หรือ texture

8. EnableServoLoop(BOOL bEnable)

Function Prototype:

virtual void EnableServoLoop(BOOL bEnable);

Function Description:

HapticView จะเรียกใช้งานฟังก์ชั่นนี้ในการ start และ stop servoloop ต้องทำการ override function นี้ในการเรียก gstScene::startServoLoop() หรือ gstScene::stopServoLoop() ในส่วน instance ของ gstScene และค่าของ bEnable จะบอกว่าจะให้ servoloop ทำการ start หรือ stop

9. QueryPHANToMNames()

Function Prototype:

virtual LPCSTR* QueryPHANToMNames();

Function Description:

ในขณะที่มีการกำหนดค่าเริ่มต้นของ view HapticView ต้องการทราบถึง PHANToM ที่ใช้อยู่ ต้องการที่จะ reset ถ้าไม่ต้องการจะเรียก StartProgram ภายหลังที่มีการสร้าง view เสร็จ ถ้าไม่เป็นเช่นนั้นจะแสดงรูปของการกำหนดตำแหน่งของ PHANToM ส่วนฟังก์ชั่นนี้นั่นใช้ในการเรียกชื่อของอุปกรณ์ที่ได้ทำการตั้งชื่อเอาไว้ HapticView นั้นจะตั้งชื่อได้แค่ 2 แต่ในส่วนของ GHOST นั้นจะสามารถกำหนดได้ถึง 4

```
LPCSTR *MyHapticView::QueryPHANToMNames()
{
static LPCSTR apszNames[] = {
"Default PHANTOM",
NULL
};
return apszNames;
}
```

10. QueryCursorPos(double* pX, double* pY, double* pZ)

Function Prototype:

virtual BOOL QueryCursorPos(double* pX, double* pY, double* pZ);

Function Description:

เป็นความต้องการของ HapticMouse ในการเรียกดูค่าตำแหน่งปัจจุบันของ PHANToM ในพิกัดของโปรแกรมซึ่งตำแหน่งนี้นั้นต้องการในการแปลงจาก 3D ไปเป็นพิกัด 2D ของ mouse ซึ่งฟังก์ชั่นนี้จะถูกเรียกใช้เมื่อเกิดเหตุการณ์ OnPhantomLeave()

3.3 ระบบย่อยส่วนการแสดงผลกราฟิก

ในการจำลองการทำงานของระบบเพื่อให้มีความสมจริงมากขึ้นในการทำงานรวมถึงมีรูปภาพหรือลักษณะของวัตถุเพื่อใช้ในการแสดงผล และการแสดงผลลัพธ์ให้กับผู้ใช้รับทราบถึงสถานะที่เปลี่ยนไปของระบบ ซึ่งต้องอาศัยกลไกของ GHOST SDK เพื่อใช้ในการแสดงผล ซึ่งในส่วนของ GHOST SDK นั้นมีส่วนในการสนับสนุนกราฟิกไลบารี่อยู่แล้วนั่นคือ GHOST SDK ซึ่งได้มีการพัฒนาเพื่อให้ผู้พัฒนาระบบสามารถที่จะรวมเอากราฟิกเข้าไปแสดงผลในส่วนของสภาพแวดล้อมการสัมผัส การแสดงผลนั้นจำเป็นต้องมีการรับส่งข้อมูลระหว่าง Haptic Process กับ Graphic Process เพื่อให้การทำงานประสานสัมพันธ์กัน ซึ่ง GHOST SDK ได้ทำการจัดเตรียมการทำงานไว้ให้นั่นคือ Graphic Callback รวมถึงมีส่วนของของการ Redraw ซึ่งจะทำการ Redraw ประมาณ 30 Hz ซึ่งการจัดเตรียมการรองรับการทำงานของ OpenGL นั่นเป็นหน้าที่ของ GHOST SDK เป็นผู้จัดการให้ทำให้ผู้พัฒนาระบบไม่ต้องทำการกำหนดการแสดงผลเองทำให้การพัฒนาระบบมีความสะดวกมากยิ่งขึ้น แต่อาจจะไม่ยืดหยุ่นนักซึ่งตัว SDK เองได้มีส่วนในการยินยอมให้ผู้พัฒนาระบบสามารถที่จะกำหนดสถานะของกราฟิกเองแต่ ก็ไม่ค่อยจะสะดวกมากนักสืบเนื่องจากเรื่องการกำหนดมุมมอง กำหนดแสงต่างๆ ผู้พัฒนาต้องทำการกำหนดเองทั้งหมด ซึ่ง

ผู้เขียนคิดว่าสร้างความยุ่งยากให้มากกว่า แต่ถ้าต้องการความยืดหยุ่นหรือผู้พัฒนามีความชำนาญนับว่าเป็นตัวเลือกที่น่าสนใจในการพัฒนา แต่การพัฒนาร่วมกับ GHOST SDK4.0 นับว่าไม่มีปัญหาในการทำงานร่วมกับ OpenGL

3.3.1 แนวคิดระบบย่อยในส่วนของกราฟิก

การแสดงผลในส่วนของการฟิก จะมีส่วนประกอบไปด้วยส่วนหลักๆ คือ ในส่วนของ HIP หรือตัวเคอร์เซอร์ การแสดงผนังซึ่งบอกขอบเขตของสภาพแวดล้อม การแสดงผลในส่วนของกระดาษ รวมถึงการแสดงผลเมื่อส่วนของ HIP ได้ทำการแตะกับกระดาษเพื่อแสดงการวาดภาพ ส่วนของ HIP นั้นจะมีลักษณะเป็นทรงกลมเคลื่อนที่ตามที่เราขยับอุปกรณ์แรงสัมผัส ส่วนผนังนั้นจะอยู่กับที่ไม่สามารถเคลื่อนไหวได้ ส่วนของกระดาษและส่วนของการจำลองการวาดนั้นจะใช้หลักการของ Texture mapping และ Sub Texture mapping รวมถึงจะมีการปรับเปลี่ยนขนาดของหน้าต่างเมื่อผู้ใช้มีการปรับเปลี่ยนขนาด

3.3.2 การแสดงผลกราฟิกในส่วนของ Haptic Interface Point

หลักการทำงานของการแสดงผลของ HIP นั้นกล่าวคือ

1. สร้างวัตถุทรงกลม
2. เรียกตำแหน่งของ PHANToM
3. ย้ายวัตถุนั้นไปที่ตำแหน่งของ PHANToM ที่ได้มา

```
Pseudo Code สำหรับการเรียกตำแหน่งและการย้ายวัตถุ
        gstPoint cursorPos;
        ((gstPHANToM *) phantom)->getPosition_WC(cursorPos);
        glTranslatef(cursorPos.x(), cursorPos.y(), cursorPos.z());
End
```

รูปที่ 3.1 Pseudo Code สำหรับการเรียกตำแหน่งและการย้ายวัตถุ

3.3.3 การแสดงผลกราฟิกในส่วนของผนัง

ผนังทั้ง 4 ด้านนั้นจะเป็นตัวกำหนดขอบเขตของการเคลื่อนที่ของ HIP ซึ่งในการสร้างผนังนั้นจะสัมพันธ์กับการกำหนดขอบเขตของ haptic ซึ่งจะกล่าวถึงต่อไปในการสร้างผนังจะประกอบไปด้วย 4 ส่วนคือ ด้านขวา ด้านซ้าย ด้านล่าง และด้านหลัง จะใช้ OpenGL ในการวาดผนัง

```
// Create list for  right wall
        glBegin(GL_QUADS);
                glNormal3f(-1.0, 0.0, 0.0);
                 glVertex3f(0.5, -0.5, -0.5);          // right bottom back
                glVertex3f(0.5, -0.5, 0.5);           // right bottom front
                glVertex3f(0.5, 0.5, 0.5);            // right top front
                glVertex3f(0.5, 0.5, -0.5);           // right top back
        glEnd();
```

รูปที่ 3.2 ตัวอย่างการวาดผนังโดยใช้ OpenGL

3.3.4 การแสดงผลกราฟิกในส่วนของการแสดงผลกระดาษ

ในการแสดงผลรูปกระดาษนั้นใช้หลักการของ Texture mapping ซึ่งอ่านภาพจากแฟ้มข้อมูลขึ้นมาจัดเก็บไว้ใน buffer แล้วทำการ map ภาพนั้นลงไปที่รูปโพลิกอนแล้วทำการแสดงผลเพื่อให้ผู้ใช้สามารถเห็นได้ ต่อไปเป็นการแสดง Pseudo Code ของการทำ Texture mapping ซึ่งหลักการทำงานคือ จะทำการโหลดรูปเข้ามาในหน่วยความจำแล้วทำการสร้าง texture ขึ้นมาจากรูปที่โหลดเข้ามา ก่อนที่จะทำให้เป็น texture ต้องมีการกำหนด parameter บางอย่างเพื่อกำหนดคุณลักษณะของ texture แล้วทำการ map texture ที่ได้มากับรูปโพลิกอนในที่นี้คือรูปสี่เหลี่ยม

Pseudo Code สำหรับการทำ Texture

```
TextureImage =LoadBMP(PICTURE)
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
glTexImage2D(GL_TEXTURE_2D, 0, 3, TextureImage->sizeX, TextureImage->sizeY, 0,
GL_RGB, GL_UNSIGNED_BYTE, TextureImage->data);
End
```

รูปที่ 3.3 Pseudo Code สำหรับการทำ Texture

Pseudo Code สำหรับการโหลดภาพ Bitmap

```
        FILE *File=NULL;                    // File Handle
        File=fopen(Filename,"r");           // Check To See If The File Exists
        auxDIBImageLoad(Filename);          // Load The Bitmap
End
```

รูปที่ 3.4 Pseudo Code สำหรับการโหลดภาพ Bitmap

```
Pseudo Code สำหรับการทำ Texture Mapping
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f);  glVertex3f(-0.5, -0.5, -0.5);      // back bottom left
        glTexCoord2f(1.0f, 0.0f);  glVertex3f(0.5, -0.5, -0.5);       // back bottom right
        glTexCoord2f(1.0f, 1.0f);  glVertex3f(0.5, 0.5, -0.5);        / back top right
        glTexCoord2f(0.0f, 1.0f);  glVertex3f(-0.5, 0.5, -0.5);       // back top left
        glEnd();
End
```

รูปที่ 3.5 Pseudo Code สำหรับการทำ Texture Mapping

3.3.5 การแสดงผลกราฟิกในส่วนของการแสดงการวาดภาพ

หลักการทำงานในส่วนของการแสดงการวาดภาพนั้นจะใช้หลักการทำ subtexture mapping แล้วทำการวาดภาพนั้นใหม่ไปเรื่อยๆ เมื่อมีการวาดภาพใหม่ก็จะทำการปรับปรุงภาพเก่าพร้อมทั้งเรียกภาพใหม่ขึ้นมาแสดง กระบวนการในการเรียกโหลดภาพนั้นเหมือนดังที่กล่าวไปแล้ว ส่วนการเรียกภาพซึ่งเป็นตัวอย่างของสีเทียนนั้นจะเป็นลักษณะของรูปภาพขนาดเล็กมาทำการปรับปรุงภาพขนาดใหญ่ ซึ่งต้องทำการรับค่าตำแหน่งของอุปกรณ์เพื่อให้ทราบถึงพิกัดที่ต้องนำเอาพื้นผิวไปทำการ map ลง

```
Pseudo Code  LoadSubTexture(int cx, int cy)
        AUX_RGBImageRec *Image = auxDIBImageLoad(SUBPICTURE);
        glBindTexture(GL_TEXTURE_2D, texture);
        glTexSubImage2D(GL_TEXTURE_2D, 0, cx, cy, Image->sizeX,
                Image->sizeY, GL_RGB, GL_UNSIGNED_BYTE, Image->data);
End
```

รูปที่ 3.6 Pseudo Code LoadSubTexture

3.3.6 สรุป

ในส่วนของการแสดงผลกราฟิกในโครงการนี้ก็ได้อธิบายพร้อมตัวอย่างรหัสโปรแกรม ในการแสดงกราฟิกนั้นจะประกอบด้วยส่วนหลักๆคือ ส่วนของการแสดงผล HIP การแสดงขอบเขตของโปรแกรม การแสดงผลกระดาษจำลอง รวมถึงการจำลองการวาดภาพ การแสดงการทั้งหมดจะถูกเรียกใช้งานที่ฟังก์ชั่น UpdateGraphics ซึ่งจะถูกเรียกใช้ประมาณ 30 Hz ทำให้การทำงานของกราฟิกโพรเซส สัมพันธ์กันกับ Haptic โพรเซส

3.4 ระบบย่อยส่วนการจำลองการสัมผัส

การจำลองการสัมผัสมีหน้าที่ในการควบคุม Haptic โพรเซส การสร้างและจัดการกับ Haptic Scene Graph การแสดงกราฟิกในส่วนของ Haptic Scene Graph การจำลองแรงต้านเมื่อทำการเขียนสีเทียน รวมถึงการรับรู้เมื่อมีการสัมผัสกับวัตถุ ในการเขียนโปรแกรมที่ดำเนินในโครงการนี้นั้นเป็นการเพิ่มเติมในส่วนของ gstShape เพื่อแสดงลักษณะรูปทรงของกระดาษ และ gstEffect ที่ใช้ในการแสดงการสัมผัสรวมถึงแรงต้าน

3.4.1 ภาพรวมของการใช้งาน GHOST SDK

ดังที่กล่าวไปแล้วในบทที่ 2 นั้นในส่วนนี้จะกล่าวถึงการโปรแกรมด้วย GHOST SDK เพื่อเป็นแนวทางในการพัฒนาต่อไป

3.4.1.1 ภาพรวมของโครงสร้างการโปรแกรม

ในการสร้าง Haptic Application ด้วย GHOST SDK มีโครงสร้างการโปรแกรม คือ

1. สร้าง Application ตามปกติ
2. สร้าง Haptic Scene Graphs และกำหนด Graphic Callbacks หรือ Event Callbacks
3. สร้าง Scene Node และกำหนด Haptic Scene Graphs ที่ใช้
4. ควบคุม Haptic Process ผ่าน Scene Node
5. เมื่อต้องการข้อมูลของ Haptic Process ให้เรียกผ่าน Scene Node เพื่อให้ Haptic Process เตรียมข้อมูลให้พร้อมก่อน และจะเรียกกลับผ่าน Callback ที่กำหนดใน Haptic Scene Graphs
6. เมื่อเลิกใช้ ทำการยกเลิกการจองทรัพยากรต่าง ๆ

รูปที่ 3.7 โครงสร้างการโปรแกรมด้วย GHOST SDK

3.4.1.2 การสร้าง Haptic Scene Graphs

GHOST SDK ใช้การสร้าง Haptic Scene Graphs เพื่อกำหนด Haptic Environment โดยการนำ Node ต่าง ๆ มาสัมพันธ์กับแบบ Tree โดยมี Node บนสุดทำหน้าที่เป็น Root ของ Haptic Scene Graph มี Node ภายในที่ทำหน้าที่กำหนดกลุ่มหรือเพื่อสืบทอดลักษณะ, พฤติกรรมต่าง ๆ เป็นลำดับชั้น มี Node ปลายสุดที่ทำหน้าที่แทนตัววัตถุ หรือการตอบสนองต่อแรงในรูปแบบต่าง ๆ ที่ต้องการ Simulation

ประเภทของ Node แบ่งตามประเภทของ Class ซึ่งมีลักษณะหน้าที่ต่าง ๆ กัน คือ

1. Geometry Node Class

ประกอบด้วย gstCube, gstCone, gstCylinder, gstSphere, gstTorus และ gstTripolyMeshHaptic ซึ่งรูปทรงทั้งหมดนี้ derived มาจากคลาส gstShape ทำหน้าที่แทนรูปทรงตามชื่อ และกำหนดลักษณะต่าง ๆ ของรูปทรงนั้น ตัวอย่าง คือ

```
gstCube* cube = new gstCube;
cube ->setGraphicsCallback(cubeCB);
cube ->setTranslate(0,50,0);
cube ->setWidth(2.0);
cube->setHeight(5.0)
cube ->setLength(0.3);
```

รูปที่ 3.8 ตัวอย่างการโปรแกรม Geometry Node

เป็นการสร้างลูกบาศก์ขนาดความกว้าง 2.0 ความสูง 5.0 ความลึก 0.3 ที่จุด 0,50,0

2. Hierarchical Node Class

คือ gstSeparator ทำหน้าที่รวมกลุ่ม Node เพื่อการจัดการที่เป็นกลุ่มเดียวกัน เช่น เคลื่อนย้าย, หมุน, ปรับขนาด ตัวอย่าง คือ

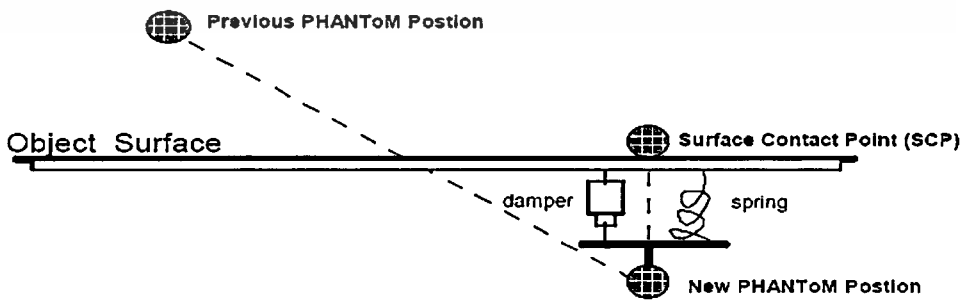
```
gstSeparator *root = new gstSeparator;
gstSeparator *rootPhantom = new gstSeparator;
gstSeparator *rootHaptic = new gstSeparator;
root->addChild(rootPhantom);
root->addChild(rootHaptic);
rootHaptic->addChild(sphere);
rootHaptic->addChild(cube);
rootHaptic->setRotate(gstVector(0.0,0.0,1.0),M_PI/2.0);
```

รูปที่ 3.9 ตัวอย่างการโปรแกรม Hierarchical Node

เป็นการสร้าง Root ของ Haptic Scene Graphs คือ root และสร้าง Tree ภายใต้ root โดยแยกเป็น rootPhantom และ rootHaptic จากนั้นรวมกลุ่ม sphere และ cube ภายใต้ rootHaptic แล้วหมุนทั้งกลุ่มรอบแกน z ทิศทวนเข็มนาฬิกา 90 องศา

3.4.1.3 Haptic Interface Device Node Class

คือ gstPHANToM และ gstPHANToM_SCP ทำหน้าที่แทนตัวเครื่องมือ เพื่อให้ข้อมูลตำแหน่งและสถานะของตำแหน่งจริงและตำแหน่งสัมผัสพื้นผิว Surface Contact Point (SCP) (SCP เกิดเมื่อสัมผัสกับรูปทรง) ตามลำดับ



รูปที่ 3.10 ตำแหน่งสัมผัสพื้นผิว

3.4.1.4 การสร้างสภาพแวดล้อมของ Haptic

หลังจากที่ได้ทราบถึงโหนดคลาสแบบต่างๆ แล้วขั้นตอนต่อไปเป็นการสร้างสภาพแวดล้อมการสัมผัส ขั้นตอนพื้นฐานมีอยู่ 2 วิธี คือ

1. สร้าง Scene Graph ซึ่งประกอบไปด้วย รูปทรงหรือวัตถุต่างๆ separator และ dynamic node

2. นำส่วนของ Phantom Haptic interface เข้าไปยังใน scene

สภาพแวดล้อมการสัมผัสแบ่งได้เป็น 2 ชนิดคือ แบบ static กับแบบ dynamic กล่าวคือ

1. Static Haptic Environment

เป็นลักษณะของสภาพแวดล้อมที่วัตถุไม่สามารถเคลื่อนที่ได้ เมื่อมีการสัมผัสหรือตอบสนองต่อการแรงกระทำ

2. Dynamic Haptic Environment

เป็นลักษณะของสภาพแวดล้อมที่วัตถุสามารถตอบสนองต่อการกระทำของ PHANToM ได้ ซึ่งจะต้องเพิ่มในส่วนของ gstDynamic เข้าไปตัวอย่างเช่น

```
gstCube *myCube = new gstCube;
        myCube->setWidth(50);
        myCube->setHeight(50);
        myCube->setLength(20);
gstButton *myButton = new gstButton;
        myButton->addChild(myCube);
```

รูปที่ 3.11 ตัวอย่างโปรแกรม Dynamic Haptic Environment

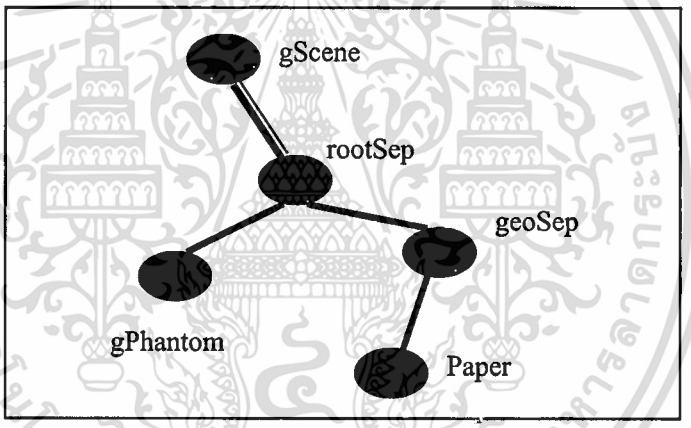
จากรูป myCube เป็นลักษณะของวัตถุที่ static แต่เมื่อทำการนำเอา myCube เข้าไปใส่ภายใต้ myButton ซึ่งเป็นลักษณะของ Dynamic คือปุ่มเป็นผลให้สามารถที่จะกดปุ่มได้เป็นลักษณะของ Dynamic

3.4.2 แนวคิดในการพัฒนาส่วนจำลองแรงสัมผัส

ในการจำลองแรงสัมผัสจะประกอบไปด้วย 2 ส่วน คือส่วนที่จำลองแรงหนืดของกระดาษ และเมื่อได้มีการวาดรูปลงไปบนพื้นผิวของกระดาษแล้วทำการวาดซ้ำอีกครั้งหนึ่งนั้นความหนืดของกระดาษต้องลดลง ซึ่งการกระทำการนี้จะต้องไปเพิ่มเติมในส่วนของ gstEffect และอีกส่วนหนึ่งนั้น จะทำการสร้างแรงสัมผัสการเคลื่อนที่เมื่อผู้ใช้งานได้ทำการสัมผัสกับวัตถุ ซึ่งต้องไปเพิ่มเติมในส่วนของ gstShape

3.4.3 การออกแบบ Haptic Scene Graph

Scene graph เป็นการออกแบบเพื่อใช้ในการจัดกลุ่มจัดการกับกลุ่มของวัตถุต่างๆ เพื่อใช้ในการทำงานของ Haptic โพรเซส ดังรูปที่ 3.12 เป็นการออกแบบ Haptic Scene Graph



รูปที่ 3.12 Haptic Scene Graph

จากรูปเป็นการออกแบบ Haptic Scene Graph ซึ่งประกอบไปด้วย gScene ซึ่งเป็น Scene ของระบบและทำการกำหนด Root ของ Scene นั่นคือ rootSep ซึ่งเป็น gstSeparator ส่วน gPhantom เป็นส่วนของ PHANToM และอีกฝั่งหนึ่งเป็นในส่วนของ geoSep ซึ่งเป็นส่วนของกราฟิกที่แสดงใน Scene Graph นั่นคือ paper

ต่อไปจะเป็นการแสดงในส่วนของการโปรแกรมในการสร้าง Scene Graph ที่พัฒนาขึ้นในโครงการซึ่งอ้างอิงมาจากรูปการออกแบบ Scene Graph ที่ได้กล่าวถึงไปแล้ว

```
gScene = new gstScene();
 rootSep = new gstSeparator();
        gScene->setRoot(rootSep);
        gPhantom = new gstPHANToM(PHANTOM_NAME, PHANToMReset);
        gPhantom->setGraphicsCallback(phantom_graphics_callback, NULL);
 rootSep->addChild(gPhantom);
 paper = new CPlane(workSpaceBounds);
        paper->setTouchEventCallback(contact_event_callback, NULL);
geomSep = new gstSeparator;
rootSep->addChild(geomSep);
geomSep->addChild(paper);
```

รูปที่ 3.13 การโปรแกรมHaptic Scene Graph

3.4.4 การกำหนดขอบเขตของการทำงาน

ในการโปรแกรมนั้นสิ่งที่สำคัญอีกสิ่งหนึ่งคือการกำหนดขอบเขตของการทำงานกล่าวคือการที่มีข้อกำหนดในการเคลื่อนที่ขอบอุปกรณ์ไม่ให้เกินขอบเขตที่กำหนด เพื่อให้มีพื้นที่ในการทำงานที่แน่นอน การดำเนินการ gstBoundaryCube ซึ่งเป็นคลาสในการกำหนดขอบเขตของโปรแกรมว่าจะให้อุปกรณ์เคลื่อนที่ไปได้ทิศทางใดระยะทางเท่าใดบ้าง ซึ่งการกำหนดในโครงการนี้จะใช้เป็นลักษณะของลูกบาศก์ ดังโปรแกรม

```
gstBoundaryCube *workSpaceBounds
workSpaceBounds = gPhantom->attachMaximalBoundary();
workSpaceBounds->setWidth(m_Xmax-m_Xmin);
workSpaceBounds->setHeight(m_Ymax-m_Ymin);
 workSpaceBounds->setLength(m_Zmax-m_Zmin);
workSpaceBounds->setPosition((m_Xmax+m_Xmin)/2,(m_Ymax+m_Ymin)/2,
                                (m_Zmax+m_Zmin)/2);
```

รูปที่ 3.14 การโปรแกรมการกำหนดขอบเขตของการทำงาน

จากรูปเป็นการกำหนดขอบเขตของการทำงานโดยการกำหนดความกว้าง ความยาว และความลึก ให้กับ gPhantom เริ่มต้นโดยการรับค่าสูงสุดมาคำนวณแล้วทำการกำหนดค่าดังที่กล่าวมาแล้วให้กับ gPhantom โหนด

3.4.5 การจำลองแผ่นกระดาษ

การจำลองกระดาษจะทำการจำลองของแผ่นกระดาษโดยการ Extend มาจากคลาส gstShape ซึ่งเป็นคลาสของการกำหนดคุณลักษณะของรูปทรง ในที่นี้เป็นรูปทรงของแผ่นกระดาษ ซึ่งกำหนดว่า เมื่อได้สัมผัสกับ HIP แล้วให้หยุดการเคลื่อนที่ของ HIP และแสดง PHANToM SCP การเพิ่มเติมในส่วนของ gstShape ประกอบไปด้วย การคำนวณหา Collision Detection

```
Retrieve contact state for gstPHANToM node.
Create a line segment from the previous position of the SCP to the current position, in the local coordinate
frame of the object.
If (the line segment intersects the object)
        Calculate the surface contact point, defined as the point on the surface of the object such that the
        distance between that point and the phantom position is minimized. Set the variable SCP, defined in
        gstShape, to this value in local coordinates.
        Calculate the normal of the object at the SCP. Set the Variable SCPnormal, defined in gstShape, to
        this value in Local coordinates.
        Update contact state in state array for PHANTOM
        Add the object to the collision list of the PHANTOM.
        return TRUE
else
        Update contact state in state array for PHANTOM
        return FALSE
```

รูปที่ 3.15 Pseudo code สำหรับการหา collision detection

การสร้างแรงต้านนั้นจะต้องทำการ derived มาจากคลาส gstEffect ซึ่งจะถูกเรียกใช้ก็ต่อเมื่อได้ทำการสัมผัสกับแผ่นกระดาษจำลอง ดังรายละเอียดการทำงานดังนี้

1. ตรวจสอบการชนกัน ถ้ามีการชนกันเรียกการทำงานของ gstEffect
2. ตรวจสอบว่าได้เคยมีการเขียนสีลงไปแล้วหรือยัง

if ยังไม่ได้ทำการวาดรูป

$$\vec{F_F} = \mu_{paper}.\vec{F_N} \qquad : \ \mu_{paper} = 0.3$$

else

$$\vec{F_F} = \mu_{wax}.\vec{F_N} \qquad : \ \mu_{wax} = 4.0$$

เมื่อ $\vec{F_F}$ คือ แรงต้านที่แสดงให้กับผู้ใช้

$\vec{F_N}$ คือ ทิศทางปกติของพื้นผิว

3.4.6 การจำลองลักษณะของพื้นผิวของกระดาษ

จากสมการ $\overrightarrow{M} = \overrightarrow{N_s} - \nabla h + (\nabla h.\overrightarrow{N_s})\overrightarrow{N_s}$ จะสามารถคำนวณหาพื้นผิวภายหลังจากการที่เพิ่มลักษณะของพื้นผิวดังมีวิธีการดังต่อไปนี้

1. การคำนวณหา normal vector $\overrightarrow{N_s}$ คำนวณได้จาก

$\overrightarrow{N_s}$ = (V2 – V1) X (V1 –V4)

Pseudo Code NormalVector

V1 = gstVector(x,y,z)

V2 = gstVector(x+1,y,z)

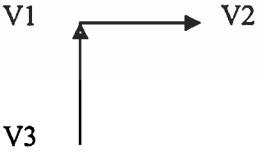
V3 = vector->sub(V2,V1)

V4 = gstVector(x,y-1,z)

V5 = vector->sub(V1,V4)

$\overrightarrow{N_s}$ = vector->cross(V3,V5)

End



รูปที่ 3.16 Pseudo Code การหา Normal vector

2. การคำนวณหา ∇h สามารถคำนวณได้จาก

```
dx = 1/3{
        Height[x+1,y]-Height[x-1,y]/2 +
        Height[x+1,y-1]-Height[x-1,y-1]/2+
        Height[x+1,y+1]-Height[x-1,y+1]/2
    }
dy = 1/3{
        Height[x,y+1]-Height[x,y-1]/2 +
        Height[x-1,y+1]-Height[x-1,y-1]/2+
        Height[x+1,y+1]-Height[x+1,y-1]/2
    }
```

∇h = dxi + dyj

เมื่อ

x คือ ค่าที่จุด IHIP สัมผัสกับกระดาษ ณ แกน x

y คือ ค่าที่จุด IHIP สัมผัสกับกระดาษ ณ แกน y

Height[x,y] คือค่าของความสว่างของรูปภาพที่จุด x ,y

3. การหาแรง $\vec{F}$ = d $\vec{M}$ เมื่อ d = HIP – IHIP

4. จะได้แรง ทั้ง 2 แรง กล่าวคือแรงที่ต้านการเคลื่อนที่ และแรงที่เกิดจากการจำลองพื้นผิว ทั้ง 2 คือแรงที่แสดงผลให้ผู้ใช้โดยดังรูปที่ 3.17

```
calcEffectForce(void *phantom)
    gstVector force;
    gstPHANToM *PHANToM = (gstPHANToM *)phantom;
    force = F + F_F ;
return force
```

รูปที่ 3.17 การโปรแกรมการสร้างแรงสัมผัส

3.4.7 สรุป

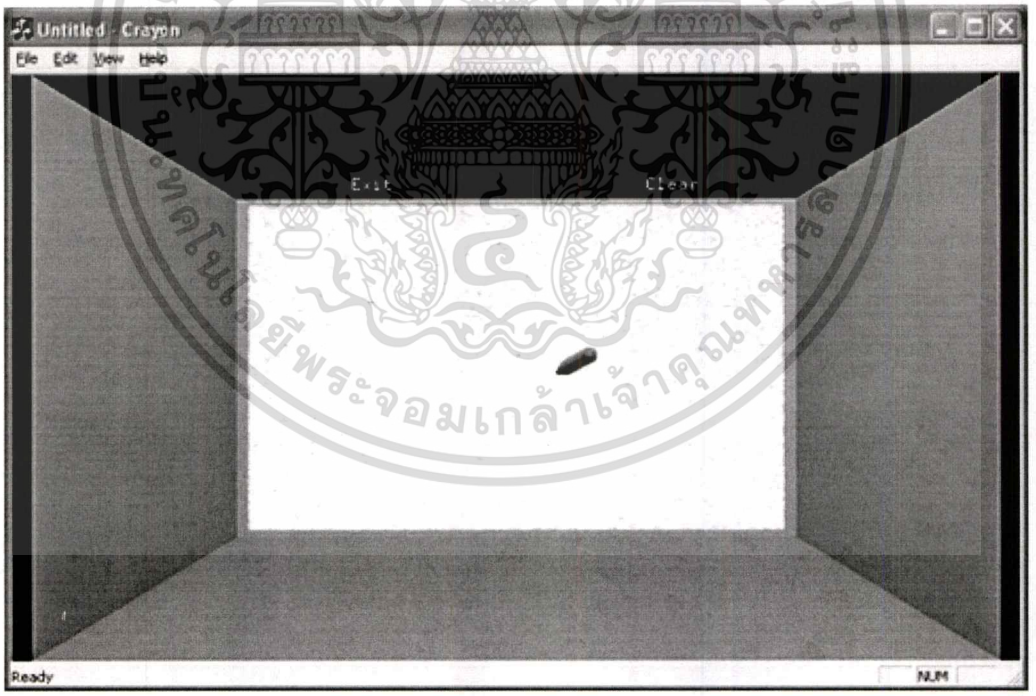
ระบบย่อยส่วนการจำลองการสัมผัสที่กล่าวมาแล้วนั้นมีด้วยกันอยู่หลายส่วน คือ ส่วนของการสร้าง Haptic Scene Graphการกำหนดขอบเขตของการทำงาน การสร้างวัตถุเพื่อกระทำการใน scene graph รวมถึงการสร้างแรงต้าน

บทที่ 4
ผลของการดำเนินโครงการ

จากการดำเนินโครงการผลที่ได้คือโปรแกรมจำลองสัมผัสของการวาดภาพด้วยสีเทียนซึ่งจะเป็นโปรแกรมจำลองการทำงานของการวาดภาพ

4.1 ภาพรวมของโครงการ

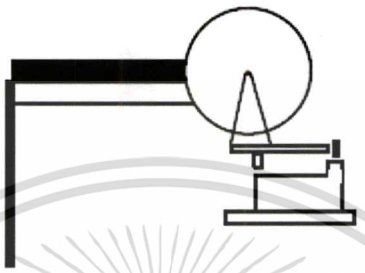
แสดงถึงภาพรวมของการดำเนินของโครงการ รูปของเค้าโครงของโปรแกรม ซึ่งจะสังเกตเห็นถึงกระดาษและผนังเมื่อต้องการวาดภาพให้เลื่อนแท่งสีเทียน ไปสัมผัสกับกระดาษหลังจากนั้นก็จะปรากฏสีไปติดลงบนกระดาษ ดังรูปที่ 4.1



รูปที่ 4.1 ภาพรวมของโปรแกรม

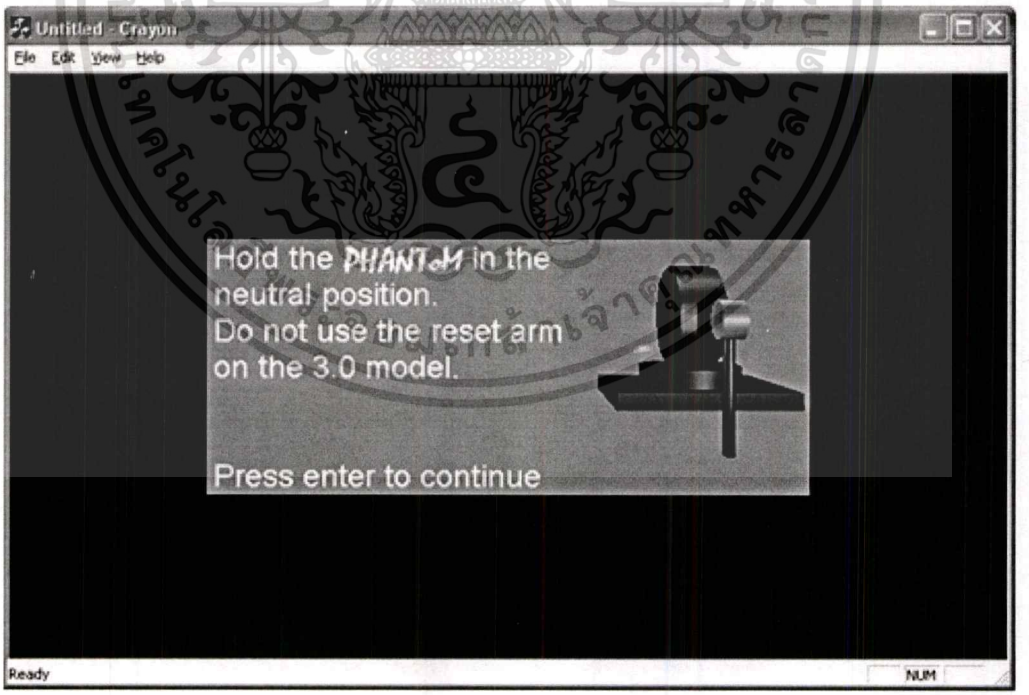
4.2 การใช้งานโปรแกรม

1. การใช้งานโปรแกรมเริ่มต้นเปิดโปรแกรมขึ้นมาจะปรากฏ ภาพให้ทำการ reset ตำแหน่งของ PHANToM ไปยังตำแหน่ง reset ซึ่งตำแหน่ง reset นั่นคือ ตำแหน่ง natural ดังรูปที่ 4.2



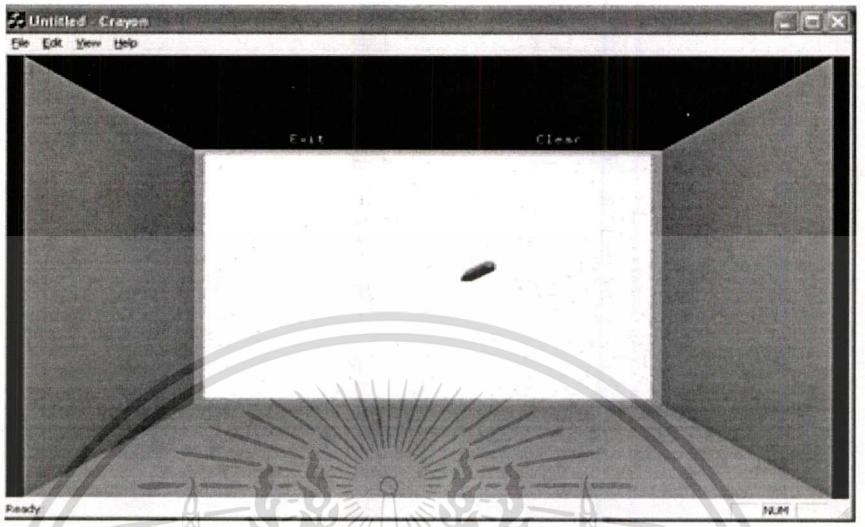
รูปที่ 4.2 ตำแหน่ง natural ก่อนการเริ่มทำงานโปรแกรม

2. เมื่อตั้งอุปกรณ์ตรงในตำแหน่งดังกล่าวแล้วให้ทำการกด Enter เพื่อเข้าสู่หน้าจอการทำงาน



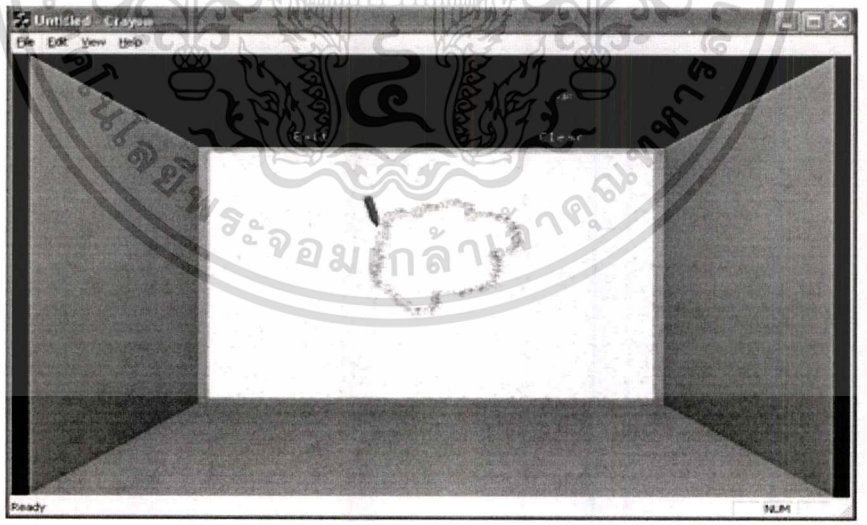
รูปที่ 4.3 รูปบอกให้ตั้งตำแหน่งของอุปกรณ์

3. เมื่อเข้าสู่หน้าจอการทำงานของโปรแกรมแล้วจะปรากฏดังรูป



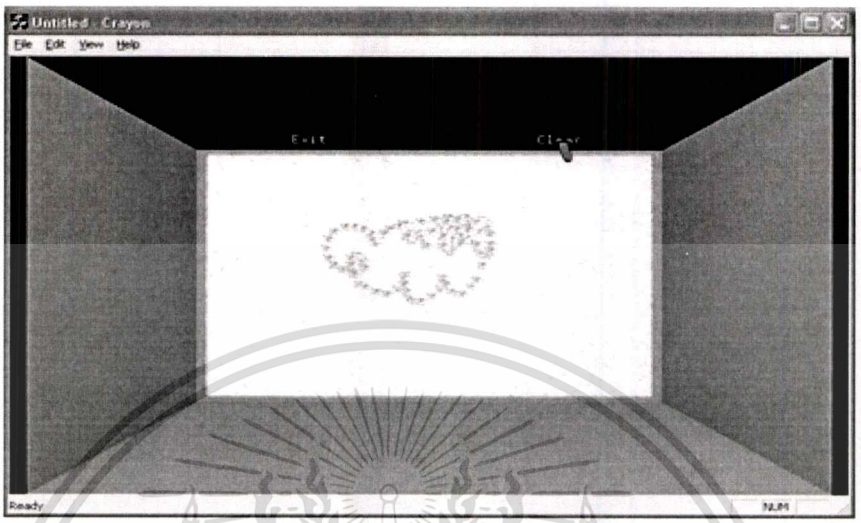
รูปที่ 4.4 หน้าจอการทำงานหลัก

4. เมื่อต้องการวาดภาพให้เลื่อนตัวสีเทียนไปยังกระดาษแล้วเริ่มทำการวาดภาพ



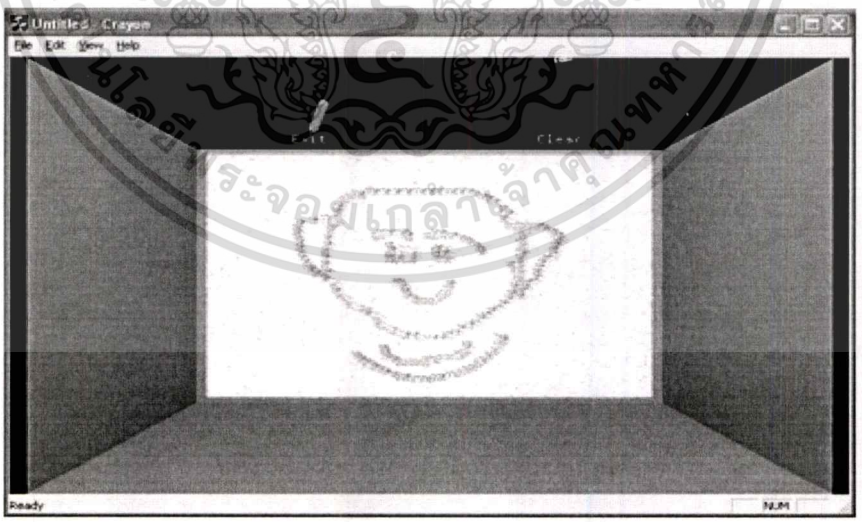
รูปที่ 4.5 เริ่มการวาดภาพ

5. ถ้าต้องการที่จะทำการลบรูปที่ได้เคยทำการวาดแล้ว ให้กดปุ่ม Clear



รูปที่ 4.6 การลบรูปที่ได้ทำการวาดแล้ว

6. ถ้าขนาดของหน้าต่างมีขนาดเล็กเกินไปสามารถขยายขนาดได้ โดยกดปุ่มขยายหน้าจอ
7. ถ้าต้องการออกจากโปรแกรมให้กดปุ่ม Exit



รูปที่ 4.7 การออกจากโปรแกรม

บทที่ 5
สรุปผลการดำเนินโครงการ

ผลการดำเนินโครงการสรุปเป็นสรุปผลการดำเนินโครงการ ข้อจำกัดของโครงการ รวมถึงข้อเสนอแนะในการพัฒนาระบบต่อไป

5.1 สรุปการดำเนินโครงการ

การดำเนินโครงการเป็นการทำงานทั้ง 2 ส่วนคือส่วนของกราฟิกในการนำรูปเข้าไปในโปรแกรม โดยการทำ texture mapping ส่วนหนึ่งเป็นการจำลองแรงสัมผัส ซึ่งทั้งหมดนี้ได้กระทำบนระบบปฏิบัติการ Windows พัฒนาด้วย Microsoft Visual C++ ร่วมกับ GHOST SDK ทำให้ได้มีโปรแกรมในการจำลองการวาดภาพขึ้นมาใช้งาน ซึ่งอาจจะไม่สมบูรณ์นักแต่ก็เพียงพอต่อการใช้งานขั้นเริ่มต้น เหมาะสำหรับผู้ที่เพิ่งเริ่มเรียนรู้ในการทำงานกับอุปกรณ์ รวมถึงผู้เขียนได้มีการเรียนรู้ในการทำงานร่วมกับอุปกรณ์จำลองแรงสัมผัส รวมถึงการโปรแกรมซึ่งนับว่าเป็นสิ่งใหม่มาก แต่โปรแกรมที่ได้ มีประสิทธิภาพเพียงพอในขั้นเริ่มต้น รวมถึงเป็นการใช้อุปกรณ์จำลองแรงสัมผัสให้แพร่หลายมากยิ่งขึ้น การวาดภาพยังเป็นการ map จากรูปยังไม่สามารถที่จะทำให้ภาพที่ได้สวยงามสมจริง แต่ก็สามารถได้เห็นถึงการเปลี่ยนแปลงของการวาดภาพ การศึกษาการโปรแกรมร่วมกับ GHOST SDK นั้นต้องใช้เวลาในการศึกษาและทำความเข้าใจอย่างมาก เพื่อให้สามารถเขียนโปรแกรมได้อย่างมีประสิทธิภาพ รวมถึงต้องมีความรู้ความเข้าใจในเรื่องของคอมพิวเตอร์กราฟิก รวมถึงเรื่อง image processing เพื่อให้การทำงานราบรื่นมากยิ่งขึ้นต้องมีความรู้ความเข้าในในเรื่องของคณิตศาสตร์พอสมควรถึงจะสามารถทำงานให้ราบรื่นและมีประสิทธิภาพ

อนึ่งในการดำเนินงานครั้งนี้ยังมีข้อบกพร่องอยู่ แต่ก็เพียงพอกับความต้องการขั้นแรกสำหรับการวาดภาพด้วยสีเทียนในส่วนของกราฟิกและส่วนของการจำลองแรงสัมผัส เมื่อเทียบกับทักษะของผู้เขียนรวมถึงเวลาที่ใช้ในการเขียนโปรแกรม

5.2 ข้อจำกัดและปัญหาของโครงการ

การดำเนินโครงการครั้งนี้ยังมีข้อจำกัดอยู่หลายประการ คือ

1. ทักษะในการเขียนโปรแกรมยังไม่ดีพอ ทำให้ล่าช้าในการเขียนโปรแกรม
2. ทักษะทางด้านคณิตศาสตร์รวมถึงฟิสิกส์ซึ่งจำเป็นมาในการศึกษาการทำงานในส่วนอื่นๆซึ่งมีเฉพาะสมการคณิตศาสตร์ซึ่งต้องใช้ความเข้าใจในการนำเอาสมการที่ได้มาทำการเขียนเป็นโปรแกรม ซึ่งในส่วนนี้ใช้เวลาอย่างมาก
3. การวาดภาพไม่สามารถที่จะลบหรือทำการเก็บข้อมูลรูปที่วาดได้
4. การจำลองแรงต้านอาจจะยังไม่สมจริงมากนัก
5. ความสมจริงของสีเทียนอาจจะยังไม่สมจริงมากนักซึ่งในการดำเนินการครั้งนี้เป็นขั้นต้นในการดำเนินงาน
6. ไม่สามารถเลือกสีของสีเทียนได้

5.3 ข้อเสนอแนะ

1. ในการดำเนินการต่อควรจะทำให้สามารถที่จะลบ จัดเก็บ รวมถึงเปลี่ยนสีของสีเทียนได้
2. อาจจะหาสมการที่มีความเหมาะสมในการจำลองแรงต้านที่สมจริงมากกว่าที่เป็นอยู่
3. สามารถที่จะแสดงภาพของสีเทียนให้มีความเสมือนจริงมากยิ่งขึ้น
4. สามารถที่จะจำลองแท่งสี รวมถึงแสดงการหดของแท่งสี

บรรณานุกรม

ยุทธนา ลีลาศวัฒนกุล. 2544 . **คู่มือการเขียนโปรแกรมและใช้งาน Visual C++ 6.0 ฉบับโปรแกรมเมอร์.** กรุงเทพฯ: อินโฟเพรส.

อุราเคน กิจพยัคฆ์. 2545. **" การพัฒนา Collaborative Networked Game โดยใช้อุปกรณ์ตอบสนองต่อแรงต้าน."** โครงการพัฒนาระบบงานวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

Douglas, Young A. 1995. **Object Oriented Programming with C++ and OSF Motif. 2nd ed.** New Jersey: Prentice-Hall.

Grigore C, BurDea. 1996. **Force and Touch Feedback for virtual reality.** New York: Wiley-Interscience.

Hill, F. S. Jr. 2001. **Computer Graphics Using OpenGL 3nd ed.** New Jersey: Prentice-Hall.

Ho,C.H. et al. 1999. **"Efficient point-based rendering techniques for haptic display of Virtual objects."** Presence: 477-491.

Minsky, M. 1995. **"Computational Haptics: The sandpaper system for synthesizing texture for a force-feedback display."** PhD thesis, Ph.D. Dissertation, Program in Media Arts and Sciences, MIT.

Sensable Technologies. 1998. **GHOST SDK Programmer's Guide Version 4.0.** Cambridge: Sensable Technologies.

ภาคผนวก

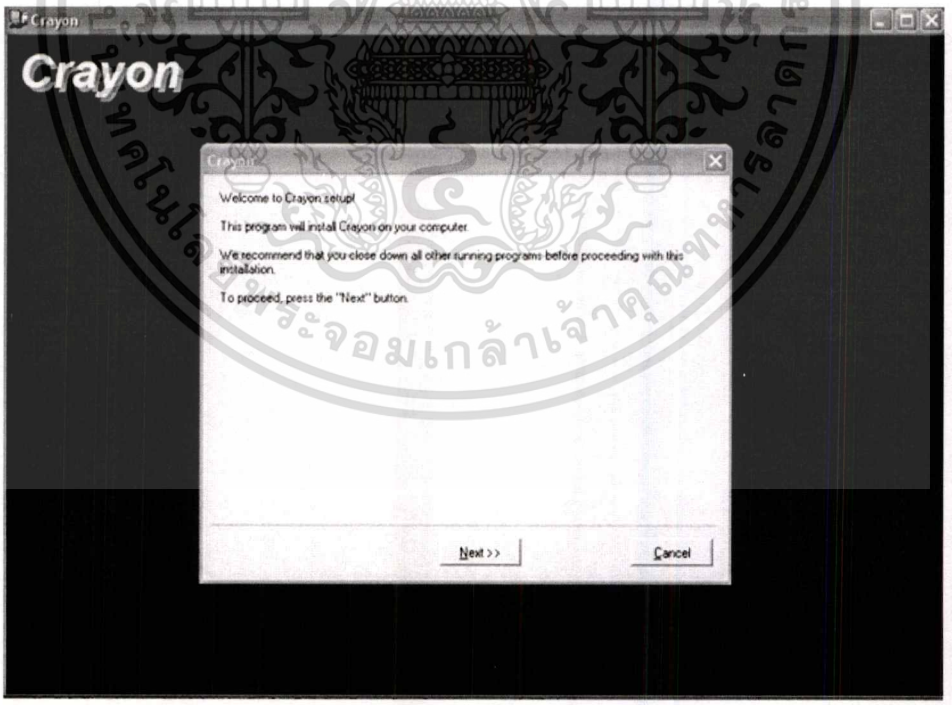
ภาคผนวก ก
การติดตั้งโปรแกรม Crayon

ก. 1 ความต้องการระบบขั้นต่ำ

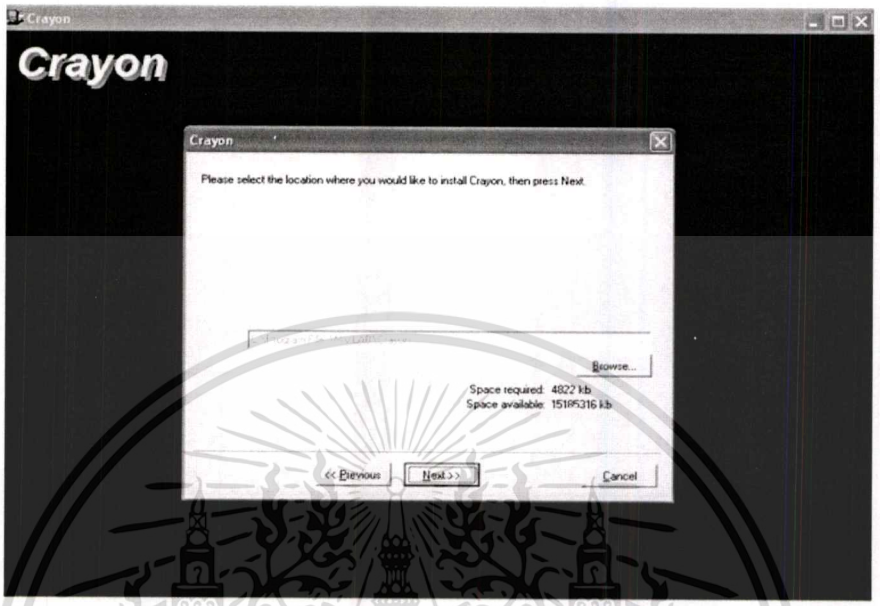
1. Pentium PC 166 MHz
2. Windows NT 4.0 with Service Pack 6.0
3. อุปกรณ์ PHANToM
4. PHANToM Device Driver Version 4.0

ก. 2 ขั้นตอนการติดตั้งใช้งาน

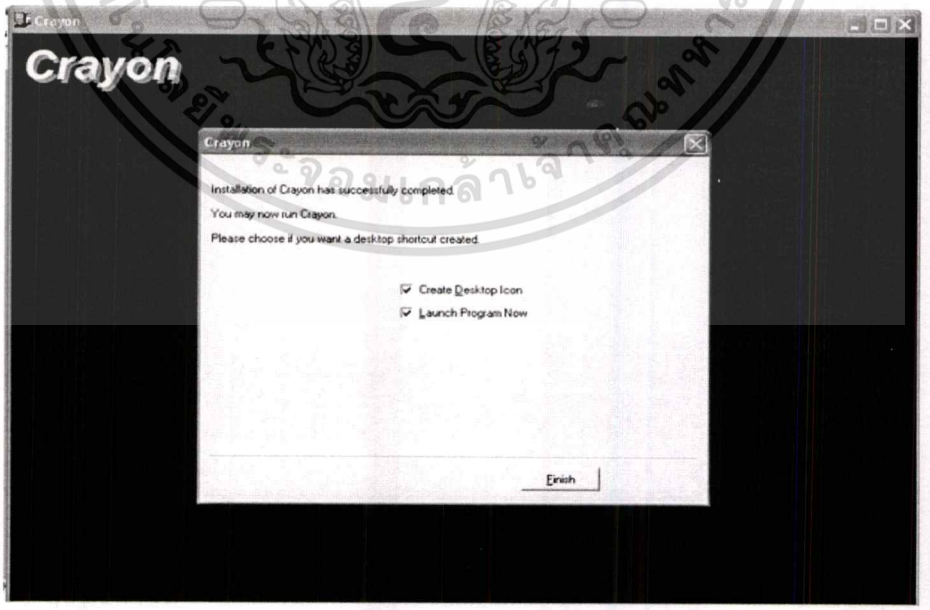
1. เปิดโปรแกรม CrayonSetup.exe เลือก Next



2. เลือก Location ที่จะทำการติดตั้ง แล้วกด Next



3. รอจนการติดตั้งสิ้นสุด ถ้าต้องการสร้าง Desktop Icon เลือกเครื่องหมายถูกที่ ช่อง Create Desktop Icon หรือถ้าต้องการเรียกโปรแกรมทำงานเลือกเครื่องหมายถูกที่ Launch Program Now สุดท้ายกดปุ่ม Finish



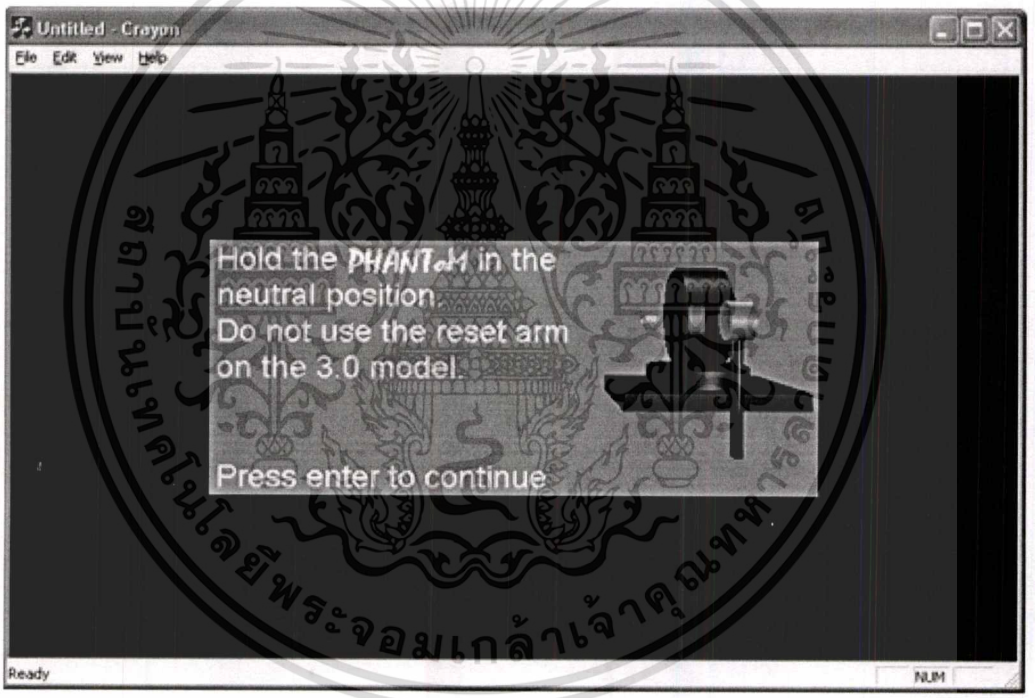
4. การเรียกใช้งานโปรแกรมคือ Start->Programs->Crayon หรือที่ Desktop Icon ถ้าสร้างไว้

ภาคผนวก ข

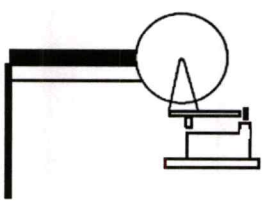
คู่มือการใช้งานโปรแกรม Crayon

ข.1 การใช้งานโปรแกรม

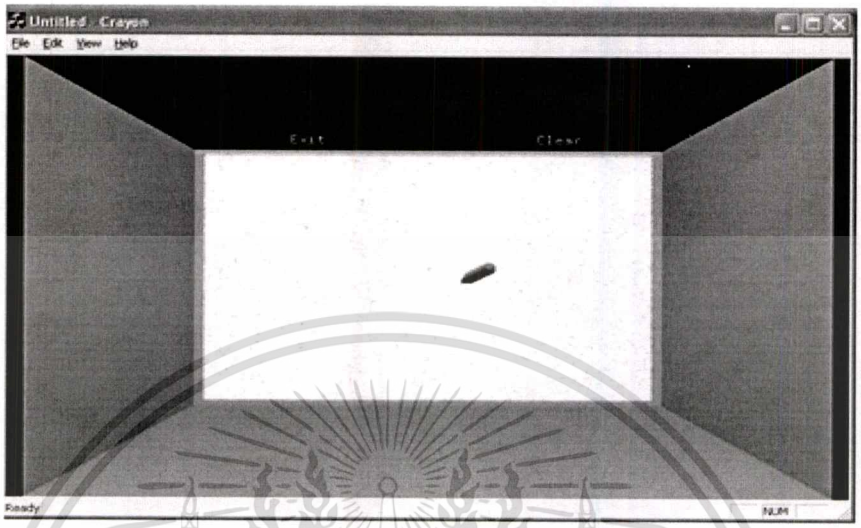
1. เปิดโปรแกรม Crayon.exe ที่ start->programs->Crayon หรือที่ Desktop Icon และทำการเปิดสวิตช์อุปกรณ์ PHANToM รวมถึงกดปุ่มการทำงานของอุปกรณ์ จะปรากฏหน้าจอดังภาพ



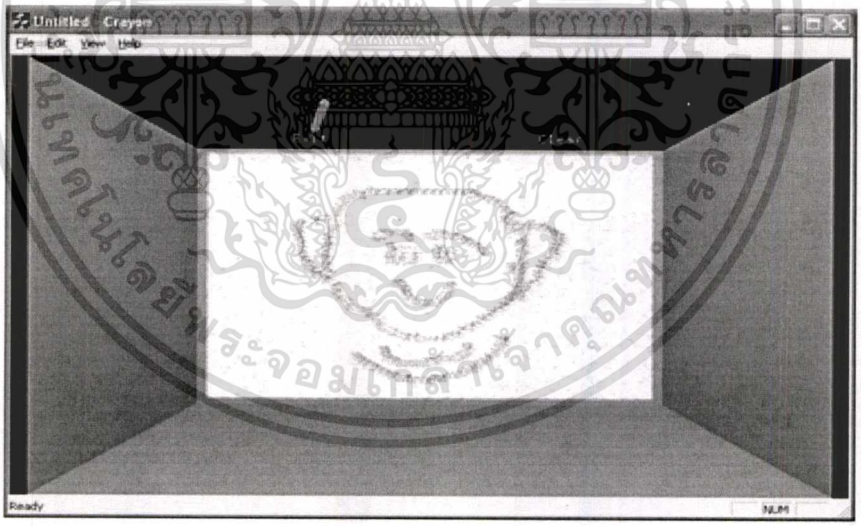
2. ให้ทำการ reset ตำแหน่งของ PHANToM ไปยังตำแหน่ง natural



3. เมื่อเข้าสู่หน้าจอการทำงานของโปรแกรมแล้วจะปรากฏดังรูป



4. ลากแท่งดินสอไปสัมผัสกับกระดาษเมื่อต้องการวาดรูป ทำการวาดภาพตามต้องการ



5. เมื่อต้องการทำการลบรูปที่เคยได้วาดแล้ว ให้ทำการกดปุ่ม Clear
6. ถ้าต้องการขยายหน้าต่างของ Windows ให้เลือกขยายหน้าต่างของ Windows
7. ถ้าต้องการออกจากโปรแกรม กดปุ่ม Exit หรือปิดที่หน้าต่างของ Windows

ประวัติผู้เขียน

ชื่อผู้เขียน	นายธีรเดช โอทกานนท์
วันเดือนปีเกิด	25 พฤศจิกายน 2523
สถานที่เกิด	ยโสธร
วุฒิการศึกษาระดับปริญญาตรี	วิทยาศาตรบัณฑิต (วิทยาการคอมพิวเตอร์)
สถานที่สำเร็จการศึกษา	มหาวิทยาลัยบูรพา
ปีที่สำเร็จการศึกษา	2545