ระบบสร้างข้อมูลแอนิเมชันของโมเดล 3 มิติ

3 D Model Animation System

โดย

อลิสา สิขัณฑกนาค

รหัส 46066222

อาจารย์ที่ปรึกษา

ดร. ธนารัตน์ ชลิดาพงศ์

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2547

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ระบบสร้างข้อมูลแอนนิเมชันของโมเดล 3 มิติ
นักศึกษา	นางสาวอลิสา สิขัณฑกนาค
อาจารย์ที่ปรึกษา	ดร. ธนารัตน์ ชลิดาพงศ์
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2547

บทคัดย่อ

ระบบแปลงข้อมูลเป็นการเคลื่อนไหวของโมเดลสามมิติ เป็นการควบคุมการเคลื่อนไหวของโมเดลสามมิติให้มีการเคลื่อนที่แบบ Real-time ตามข้อมูลที่ได้รับเข้าไป สำหรับในตัวโปรเจคนี้ จะเป็นการเขียนโปรแกรมประยุกต์ ให้สามารถรับค่าการเคลื่อนไหวและควบคุมโมเดลสามมิติที่สร้างขึ้นจากโปรแกรม Maya Version 4.5 ด้วยโปรแกรม Visual C++ เพื่อนำไปใช้ในระบบ 3D Tracking ในการทำ Motion Capture ซึ่งเกี่ยวกับการตรวจจับการเคลื่อนไหวของมนุษย์ต่อไป

Title	3D Model Animation System
Student	Miss Alisa Sikhandaganag
Advisor	Dr. Thanarat Chalidabhongse
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2004

ABSTRACT

A Control 3D Modeling with Real-Time Human Movement System can control the 3D models like human or Puppet with the real-time human movement by sending set of data to process and make the models move follow what you do. This paper describes an action for creating 3D human or Puppet modeling using Virtual Reality Modeling Languages or VRML, Maya 4.5, Microsoft Visual C++, DirectX 9.0 SDK and used for tracking and rendering in real-time 3D Motion.

กิตติกรรมประกาศ

โครงงานพัฒนาระบบและออกแบบระบบช่วยนำข้อมูลที่ได้มาแปลงเป็นการเคลื่อนไหวของโมเดล 3 มิติรูปมนุษย์และตัวการ์ตูนนี้ จะไม่สามารถสำเร็จลุล่วงไปได้ด้วยดีถ้าขาดความร่วมมือและการให้ความช่วยเหลือของบุคคลหลายท่านดังนี้

ขอขอบพระคุณ ดร.ธนารัตน์ ชลิดาพงศ์ ที่ได้ให้คำแนะนำ เป็นที่ปรึกษาในการพัฒนาโครงงาน และคอยดูแลนักศึกษาใน PIC LAB ทุกคน

ขอขอบคุณเพื่อนๆ IS 15.1 และเพื่อนๆ ใน PIC LAB ทุกคนที่คอยเป็นกำลังใจให้ในการทำงาน โดยเฉพาะ คุณ สน หาญวงศ์ และ คุณวามม์ แสงรัตนมณี ที่คอยดูแลทั้งเวลาเขียนโปรแกรมแล้วประสบปัญหา และ คำแนะนำดีๆ ในการทำโมเดลสามมิติ ให้สวยงาม จนสามารถพัฒนาระบบได้ลุล่วงไปด้วยดี

และสุดท้ายนี้ ขอขอบคุณสำหรับขนมทุกชิ้นที่บุคคลทุกคนในครอบครัวของข้าพเจ้าได้นำมาให้ยามเหนื่อยล้า และเครียดกับการทำงาน ทำให้ข้าพเจ้ามีกำลังใจที่จะพัฒนาโครงงานจนสำเร็จลงได้

อลิสา สิขัณฑกนาค

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

ตารางที่

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมาของปัญหา

ปัจจุบันการใช้คอมพิวเตอร์ในการประมวลผลภาพ(Image Processing) เพื่อทำการวิเคราะห์และวิจัยการเคลื่อนไหวของสิ่งต่างๆ เช่นมนุษย์และสัตว์ แล้วนำมาแสดงในรูปแบบสามมิตินั้น กำลังเป็นที่สนใจและเริ่มเข้ามามีบทบาทสำคัญเป็นอย่างมากโดยเฉพาะการจับการเคลื่อนไหวแบบ Real-time เพราะสามารถนำมาใช้ประโยชน์ได้หลายรูปแบบเช่น การสร้างภาพยนตร์สามมิติที่ให้ภาพและการเคลื่อนไหวเสมือนจริง แต่ใช้โมเดลโครงสร้างที่ทำจากคอมพิวเตอร์โดยการใช้คอมพิวเตอร์กราฟิก แทนผู้แสดง การสร้างแบบจำลองสนามกอล์ฟ โดยจะทำการนำภาพที่ได้จากการประมวลผล มาคำนวณหาวิถีลูกกอล์ฟ ความแรงในการตี หรือแม้แต่การสร้างเกมจำลองโดยใช้มนุษย์เป็นผู้เล่นเองก็ตาม ทั้งหมดนั้นล้วนแต่มีพื้นฐานมาจากการทำการประมวลผลรูปภาพหรือ Image Processing ทั้งนั้น

งานการศึกษาทางด้านนี้ในประเทศไทยยังคงเปิดกว้างอยู่เพราะยังมีผู้สนใจที่จะศึกษางานทางด้านนี้อย่างจริงจังเป็นส่วนน้อย ดังนั้น จึงนับว่างานทางด้าน Image Processing นั้น เป็นงานที่ค่อนข้างแปลกใหม่และน่าสนใจและน่าศึกษาเป็นอย่างมาก โดยในขณะนี้งานทางด้านการวิเคราะห์และประมวลผลเกี่ยวกับการเคลื่อนไหวของร่างกายมนุษย์ (Human Motion Analysis) ได้ถูกนำมาพัฒนาอย่างต่อเนื่อง และมีบทบาทสำคัญในการวิจัยงานทางด้าน Virtual Reality และ Computer Vision รวมทั้ง Visual Communications อีกด้วย งานวิจัยด้านการเคลื่อนไหวของร่างกายมนุษย์นั้นปัจจุบันมีการพัฒนาให้เป็น Real-time Applications โดยการทำเกมที่มีตัวละครในเกมเคลื่อนไหวได้ตามผู้เล่นนั้น ก็จัดเป็นงานที่ใช้ 3D Model และ 3D Model Analysis ในการวิเคราะห์ท่าทางการเคลื่อนไหวของตัวละครให้สมจริงด้วยโดยทั้งหมดนี้ล้วนเป็นงานประยุกต์ที่อาศัยความเสมือนจริง (virtual reality applications)

โครงงานนี้เป็นการออกแบบและพัฒนา ซึ่งเป็นส่วนหนึ่งของระบบช่วยตรวจจับการเคลื่อนไหวของร่างกายและนำข้อมูลที่ได้มาแปลงเป็นการเคลื่อนไหวของโมเดล 3 มิติรูปมนุษย์และตัวการ์ตูนต่างๆ โดยเน้นการทำงานแบบ Real-time output ระบบนี้จะเป็นการประยุกต์ใช้เทคนิคการตรวจจับการเคลื่อนไหวของมนุษย์จากกล้อง รวมทั้งเทคโนโลยีทางด้าน Computer Vision และ

Computer Graphic มาใช้ร่วมกัน โดยแนวคิดของการพัฒนาระบบนั้นสืบเนื่องมาจากความคิดในการสร้างการ์ตูนภาพยนตร์จากโมเดลสามมิติโดยจะมีการทำ Animation ให้แก่โมเดลนั้นๆ เคลื่อนไหวได้เสมือนจริง (Virtual Reality Modeling) ในรูปแบบเดิมนั้น จะต้องใช้อุปกรณ์เซนเซอร์ (Sensor) ในการตรวจจับการเคลื่อนไหวต่างๆ หลายชนิดเช่น ถุงมือเซนเซอร์ เซนเซอร์ที่สวมไว้ที่ศีรษะ เซนเซอร์ที่ติดไว้ตามร่ายกายเป็นจุดๆ ฯลฯ ซึ่งแต่ละชนิดก็มีราคาสูงมาก เป็นต้น ซึ่งในความเป็นจริงแล้ว ผู้ที่จะมีกำลังทรัพย์พอที่จะซื้อหาอุปกรณ์มาครอบครองได้นั้น ก็มีจำนวนไม่มากนัก ทำให้การสร้างภาพเคลื่อนไหวของโมเดล 3 มิติ เป็นเรื่องที่ค่อนข้างจำกัด และทำได้ในวงแคบๆ เท่านั้น จากเหตุผลดังกล่าว จึงมีแนวคิดที่จะทำการพัฒนาระบบที่สามารถจับการเคลื่อนไหวของมนุษย์แล้วนำค่าที่ได้มาแปลงเป็นผลลัพธ์แสดงการเคลื่อนไหวด้วยตัวการ์ตูน 3 มิติแบบ Real-Time ซึ่งจะใช้กล้องจับภาพเคลื่อนไหวของมนุษย์ และนำข้อมูลที่ได้มาแปลงเป็นการเคลื่อนไหวที่ใช้โมเดล 3 มิติแสดงผลด้วย ระบบนี้จะทำให้ภาพการ์ตูน 3 มิติที่ได้ออกมานั้นมีการเคลื่อนไหวสมจริง สวยงาม ทำให้ง่ายแก่การนำไปทำ Animation ของตัวการ์ตูน โดยไม่ต้องสิ้นเปลืองค่าใช้จ่ายในการซื้อเซนเซอร์ราคาแพงอีกด้วย ทั้งผู้ใช้งานยังสามารถสนุกกับการเลือกรูปแบบตัวการ์ตูนที่นำมาแสดงผลได้ พร้อมทั้งเลือก Background ประกอบฉากได้อีกด้วย

1.2 วัตถุประสงค์ของการพัฒนาระบบงาน

ในการพัฒนาระบบและออกแบบระบบช่วยตรวจจับการเคลื่อนไหวของร่างกาย และนำข้อมูลที่ได้มาแปลงเป็นการเคลื่อนไหวของโมเดล 3 มิติรูปมนุษย์และตัวการ์ตูนนี้ จะใช้การประยุกต์รวมระหว่างการใช้เทคโนโลยี Computer Vision ผสานกันกับ Computer Graphic เพื่อวัตถุประสงค์ดังนี้

1.2.1 เพื่อศึกษาหลักการและทฤษฎีทางด้าน Computer Vision ในเรื่องของการจับการเคลื่อนไหวของร่างกายมนุษย์โดยกำหนดเป็นจุดต่างๆ

1.2.2 เพื่อศึกษาการใช้โปรแกรม Microsoft Visual C++ ร่วมกับ Direct X 9.0 และ Open CV ในการสร้าง Application สำหรับการประมวลผลภาพต่างๆ และแสดงผลออกมาทางจอภาพเป็น Output

1.2.3 เพื่อศึกษาการใช้ Maya Unlimited Version 4.5 และการใช้ Plug in เพื่อ export File ที่ได้ให้เป็น File ในรูปแบบที่เราต้องการ

1.2.4 เพื่อสร้างโปรแกรมประยุกต์ที่มีความสามารถในการรับค่าการเคลื่อนไหวของมนุษย์และนำมาแปลงพร้อมทั้งแสดงผลออกทางโมเดล 3 มิติที่สร้างขึ้นแบบ ทันทีทันใด (Real-Time) ได้

1.3 ขอบเขตของการพัฒนาระบบงาน

ในการพัฒนาระบบและออกแบบระบบช่วยตรวจจับการเคลื่อนไหวของร่างกาย และนำข้อมูลที่ได้มาแปลงเป็นการเคลื่อนไหวของโมเดลสามมิติรูปมนุษย์และตัวการ์ตูนมีขอบเขตของการพัฒนาระบบงานดังต่อไปนี้

1.3.1 ยังไม่มีการเชื่อมระบบทั้งหมดเข้าด้วยกัน ในโครงงานนี้เป็นเพียงระบบส่วนเดียวคือส่วนการโปรแกรมให้โมเดลการ์ตูน 3 มิติเคลื่อนไหวได้

1.3.2 การทำโมเดลการ์ตูน มีจำนวน 3 ตัว และมีฉากหลังประกอบ 3 ฉาก

1.3.3 มีจำนวนส่วนที่ใช้ในการควบคุมการเคลื่อนไหวโดยจะใช้การแปลงค่าข้อมูลในแต่ละส่วน ที่เรียกว่า Node ประกอบด้วย 17 ส่วน ดังนี้คือ ศีรษะ คอ ลำตัวท่อนบน ลำตัวท่อนล่าง สะโพก แขนซ้ายท่อนบน แขนซ้ายท่อนล่าง มือซ้าย ขาซ้ายท่อนบน ขาซ้ายท่อนล่าง เท้าข้างซ้าย แขนขวาท่อนบน แขนขวาท่อนล่าง มือขวา ขาขวาท่อนบน ขาขวาท่อนล่าง และ เท้าข้างขวา

1.4 ประโยชน์และผลที่คาดว่าจะได้รับ

1.4.1 ได้ฝึกฝนการใช้ภาษา OOP (Object Oriented Programming) หรือภาษาเชิงวัตถุซึ่งเป็นภาษาที่นิยมกันมากในปัจจุบัน และนำไปใช้ในการพัฒนาโปรแกรมประยุกต์ที่สามารถนำไปใช้งานได้จริง

1.4.2 ฝึกฝนการสร้างโมเดล 3 มิติจากโปรแกรมสร้างโมเดล 3 มิติที่กำลังเป็นที่นิยมกันอย่างกว้างขวาง คือ Maya และสามารถนำไปใช้ในการพัฒนาการสร้างโมเดล 3 มิติเสมือนจริงได้

1.4.3 เรียนรู้ทฤษฎีทางด้าน Computer Graphic และ Computer Vision และนำไป ประยุกต์ใช้ในการพัฒนาโปรแกรมต่อไปในอนาคต

1.5 เครื่องมือที่ใช้ในระบบ

เครื่องคอมพิวเตอร์ส่วนบุคคล (PC) โดยมีข้อกำหนดคุณลักษณะดังนี้

- Pentium 4 Processor 1.8 GHz
- 1024 MB DDR-RAM
- 80 GB Harddisk ATA 5200 RPM
- RADEON9600 64MB Soundcard
- CD-ROM 40X

ซอฟต์แวร์ที่ใช้

- Microsoft Windows XP Professional
- Microsoft Visual C++
- DirectX 9.0 SDK
- Maya Unlimited 4.5

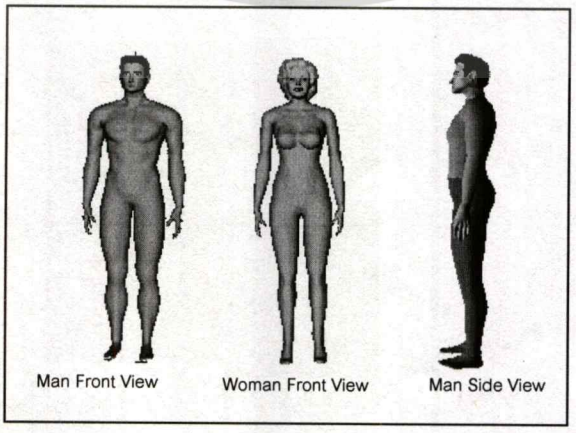
บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงหลักการและทฤษฎีรวมทั้งที่มา ที่ได้นำมาใช้ในการพัฒนาโปรแกรมประยุกต์ (Application)

2.1 ที่มาของการริเริ่มคิดทำ Humanoid Model และแนวคิดในการสร้าง Model (Hilton. 2000)

แต่แรกนั้น การทำ Humanoid Model ใช้เทคนิคหลักการจับภาพโครงสร้างสามมิติ มาใช้กับวัตถุจริงโดยมีการพัฒนาจากศูนย์วิจัยทางด้านการประมวลผลมัลติมีเดีย (The Centre for Vision, Speech and Signal Processing) นำโดยศาสตราจารย์จอห์น อิลลิงเวิธ (John Illingworth) จากมหาวิทยาลัยเซอร์เรย์ ประเทศอังกฤษ (University of Surry, England) แต่แล้วก็พบปัญหาว่าต้องใช้เวลาในการทำการประมวลผลรูปภาพนานมาก จึงทำให้งานออกมาไม่เป็น Real-time เพราะกระบวนการในการสร้าง Application นั้นทำได้ช้ามาก

ในเวลาถัดมา ได้มีการพัฒนาแบบจำลองขึ้นใหม่ซึ่งนำโดย ดร.เอเดรียน ฮิลตัน(Adiran Hilton) จากมหาวิทยาลัยเซอร์เรย์(University of Surry, England) โดยได้ทำการทดลองนำแบบจำลองโครงสร้างมนุษย์ที่ได้มีการทำไว้แล้วเป็นแบบ สาม มิติ มาทดลองใช้แทนวัตถุจริง ทำให้ค้นพบว่า การประมวลผลนั้นทำได้รวดเร็วกว่าเดิมอย่างมาก ดังนั้นแนวคิดดังกล่าวนี้ จึงเป็นก้าวแรกของการคิดค้นวิจัยให้นักวิจัยได้พัฒนารูปแบบจำลองร่างกายมนุษย์ที่มีลักษณะเหมือนจริงดังรูปที่ 2.1 ขึ้นในเวลาต่อมาจนถึงปัจจุบันนี้



รูปที่ 2.1 แสดงพื้นผิวภายนอกของ Humanoid Model

2.2 ส่วนประกอบของ Humanoid Model (Humanoid Animation Working Group. 2001)

ส่วนประกอบหลักของ Humanoid Model ดังรูปที่ 2.2 ประกอบ ด้วยส่วนต่างๆ ดังนี้

1. Segment Node

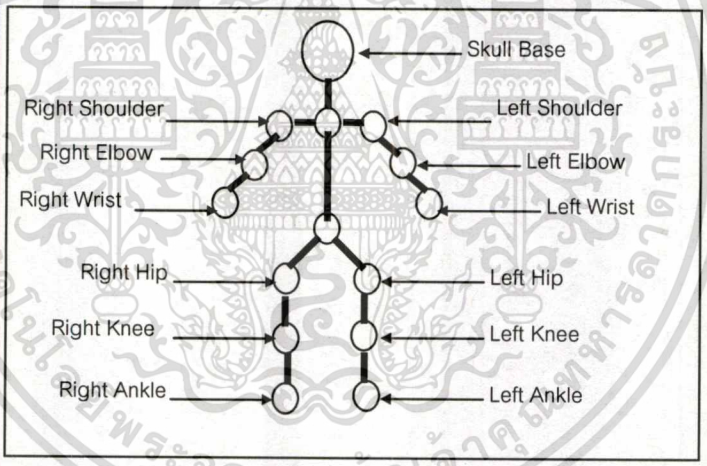
ประกอบด้วย แขน ขา มือ เท้า คอ ลำตัว ศีรษะ โดยถูกเชื่อมต่อด้วยจุดเชื่อมต่อกัน (Connected) ด้วยข้อต่อที่เรียกว่า Joint Node

2. Joint Node

ประกอบด้วย ข้อต่อคอ ข้อศอก ไหล่ ข้อมือ ข้อเท้า โดยจะใช้แทนส่วนต่างๆ ของร่างกายมนุษย์

3. Humanoid Node

จะเป็นส่วนที่ใช้เก็บข้อมูลทั่วไปของผู้ใช้ เช่น ชื่อผู้ทำระบบ วันที่สร้างแบบโครงสร้างจำลอง ข้อมูลลิขสิทธิ์ต่างๆ เวอร์ชันที่ใช้ขณะนั้น ที่อยู่ ข้อตกลง ข้อจำกัดต่างๆ ในการใช้ระบบ รวมทั้งข้อมูลที่อ้างอิงถึง Segment Node และ Joint Node อีกด้วย



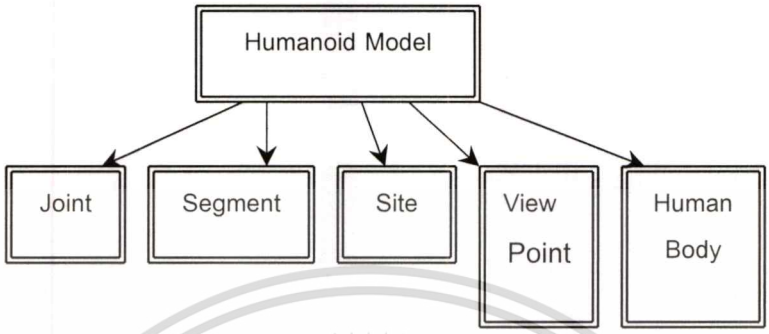
รูปที่ 2.2 แสดงส่วนประกอบของ Humanoid Model

ในการศึกษาการทำการเคลื่อนไหวของร่างกายมนุษย์นั้น ต้องคำนึงถึงข้อจำกัดต่างๆ เช่น การหมุนของแขนว่ามีขีดจำกัดได้เท่าใดตามหลักความเป็นจริง หรือการหักข้อมือ พับแขนสามารถทำได้มากที่สุดแค่ไหน การทำให้เกิดภาพเคลื่อนไหวนั้นจะนำเอาส่วนของ Joint Node มาเปลี่ยนข้อมูลที่อยู่ข้างในทำให้เกิดการเคลื่อนไหว แขน ขาขยับไปได้เป็นต้น

2.3 ประเภทของโหนด

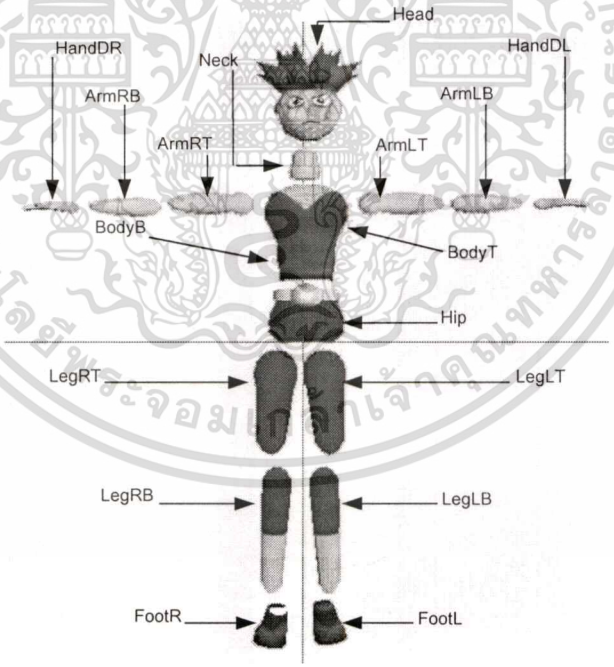
ประเภทของโหนดในการสร้าง Humanoid Model โดยทั่วไปนั้น ประกอบด้วยโหนดดังต่อไปนี้คือ Joint Node, Segment Node, Site, View Point และ Humanoid Body ซึ่งจาก

VRML นี้เอง ทำให้แต่ละโหนด ถูกกำหนดโดย Prototype Interface หรือเรียกว่า PROTO ดังรูปที่ 2.3



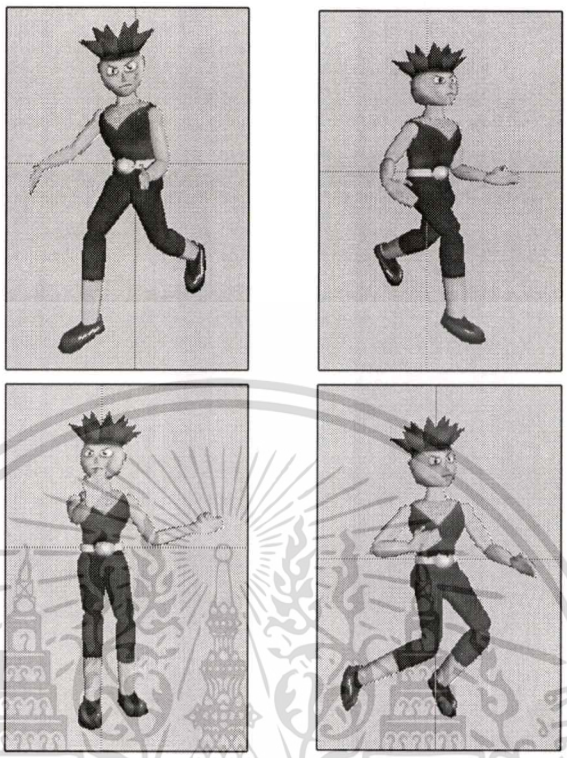
รูปที่ 2.3 แสดงส่วนประกอบ Prototype Interface

โดยในการศึกษาการเคลื่อนไหวของร่างกายมนุษย์ให้สัมพันธ์กับ Model นั้น จะต้องคำนึงถึงการระบุทิศทางของแกนในแต่ละโหนดด้วยการเคลื่อนไหวจึงจะสมจริงดังรูปที่ 2.6

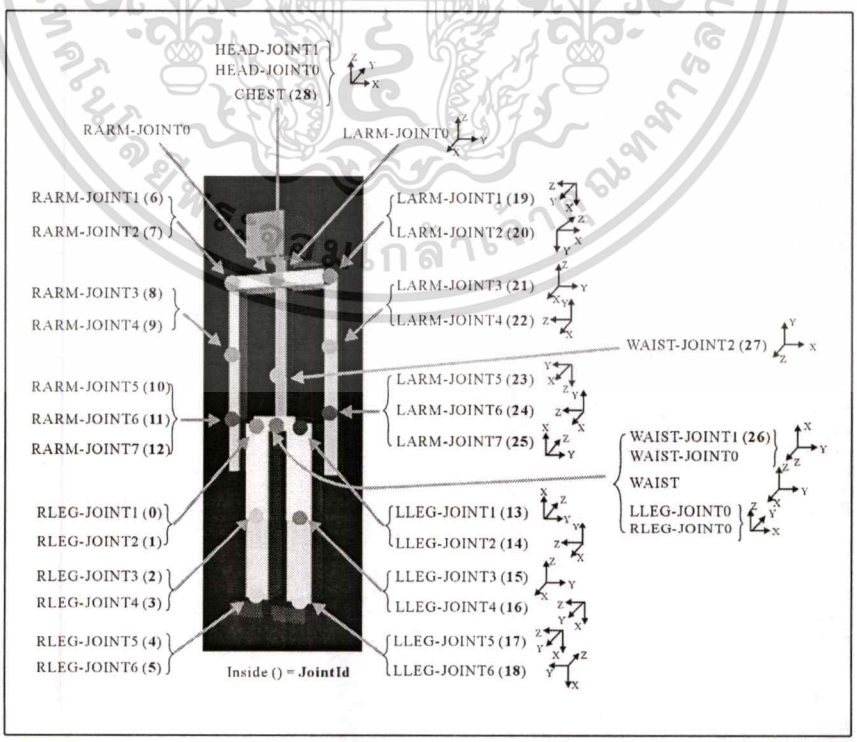


รูปที่ 2.4 แสดงส่วนประกอบของ Humanoid Model

การทำให้เกิดภาพเคลื่อนไหวนั้นจะนำเอาส่วนของ Joint Node มาเปลี่ยนข้อมูลที่อยู่ข้างในทำให้เกิดการเคลื่อนไหว แขน ขาขยับไปได้เป็นต้นดังรูปที่ 2.4 และ รูปที่ 2.5



รูปที่ 2.5 แสดงการเคลื่อนไหวสมจริงของ Humanoid Model



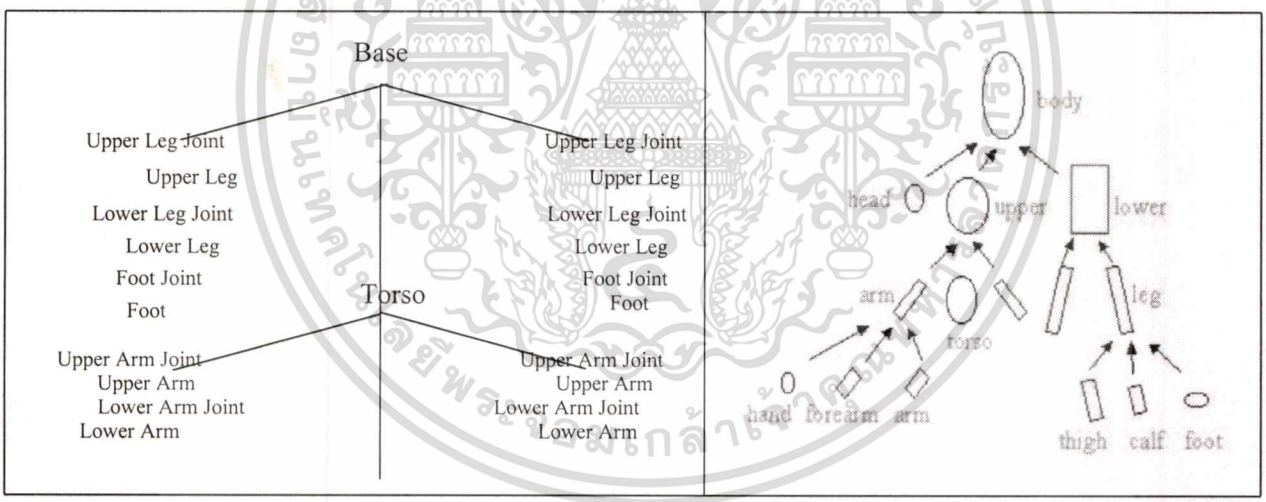
รูปที่ 2.6 แสดงการระบุทิศทางและแกนของแต่ละโหนด

2.4 มาตรฐานของการสร้าง Humanoid Model แบบ VRML

1. **Compatibility คือ ความเข้ากันได้** จะต้องไม่ผูกติดกับภาษาใดภาษาหนึ่ง ไม่ขึ้นกับภาษาสคริปต์ เช่น ไม่ใช่ว่าพอเปลี่ยนสคริปต์ก็ทำให้ใช้งานไม่ได้เป็นต้น
2. **Flexibility คือ ความยืดหยุ่นสูง** ต้องไม่มีข้อจำกัดเกี่ยวกับการใช้ Application ที่จะเข้าถึงได้
3. **Simplicity คือ ความง่าย** สามารถเขียนโปรแกรมเพื่อเพิ่มเงื่อนไขต่างๆ เข้าไปใหม่ในภายหลังได้

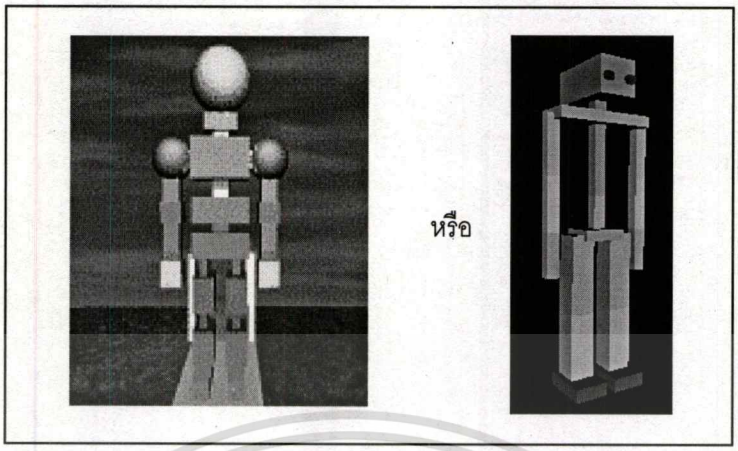
2.5 การสร้าง Humanoid Model โดย VRML

การสร้าง Humanoid Model นั้น เราสามารถจัดองค์ประกอบต่างๆ คือ Segment Node และ Joint Node ให้อยู่ในลักษณะของโครงสร้างแบบลำดับชั้น (Hierarchy Structure) ดังรูปที่ 2.7



รูปที่ 2.7 แสดงโครงสร้างแบบลำดับชั้นของ Humanoid

โดยถ้าเราจะทำ Humanoid Model อย่างง่ายนั้น ก็เพียงแค่แทนเซตของ Node ต่างๆ ด้วยรูปภาพทรงเลขาคณิต เช่น แทน Joint Node ด้วย รูปทรงกลม แทน Segment Node ด้วย รูปทรงกระบอก และแทน แกนของแบบจำลองด้วย รูปสี่เหลี่ยมลูกบาศก์ เป็นต้น หรืออาจแทนด้วยรูปทรงใดก็ได้ไม่ตายตัวแล้วแต่เรากำหนดให้ทำการวาดตามรูปที่ 2.8



รูปที่ 2.8 แสดงตัวอย่างโครงสร้างของ Humanoid ด้วยรูปทรงเลขาคณิต

การจะสร้าง Humanoid แบบสามมิติได้นั้นต้องคำนึงถึงความต้องการของระบบด้วย เพราะการสร้างรูปภาพแบบสามมิตินั้นมักจะใช้ทรัพยากรของคอมพิวเตอร์มาก จึงทำให้คอมพิวเตอร์บางตัวที่มีหน่วยประมวลผลต่ำ หรือมีความเร็ว CPU ไม่เพียงพอ จะไม่สามารถสร้างรูปสามมิติได้ โดยทั่วไปแล้ว การจะสร้างแบบจำลองสามมิตินั้น ควรจะคำนึงถึงความต้องการของระบบซึ่งเป็นสิ่งสำคัญมากด้วย

2.5.1 โครงสร้างแบบ Hierarchy ของ Humanoid Model (Takanishi. 2001)

ตารางที่ 2.1 แสดงส่วนประกอบของ Humanoid Model

ส่วน	โหนดที่	Description
#define HEAD	0	(สำหรับส่วนของศีรษะ)
#define NECK	1	(สำหรับส่วนของคอ)
#define BODYT	2	(สำหรับส่วนของลำตัวท่อนบน)
#define BODYB	3	(สำหรับส่วนของลำตัวท่อนล่าง)
#define HIP	4	(สำหรับสะโพก)
#define ARMLT	5	(สำหรับแขนซ้ายท่อนบน)
#define ARMLB	6	(สำหรับแขนซ้ายท่อนล่าง)
#define HANDL	7	(สำหรับมือซ้าย)
#define LEGLT	11	(สำหรับขาซ้ายท่อนบน)
#define LEGLB	12	(สำหรับขาซ้ายท่อนล่าง)

#define FOOTL	13	(สำหรับเท้าข้างซ้าย)
#define ARMRT	8	(สำหรับแขนขวาท่อนบน)
#define ARMRB	9	(สำหรับแขนขวาท่อนล่าง)
#define HANDR	10	(สำหรับมือขวา)
#define LEGRT	14	(สำหรับขาขวาท่อนบน)
#define LEGRB	15	(สำหรับขาขวาท่อนล่าง)
#define FOOTR	16	(สำหรับเท้าขวา)

เมื่อนำทั้งหมดมาประกอบกัน จะได้เป็นโครงร่างของมนุษย์ ตามส่วนต่างๆ ที่ได้ระบุไว้ในรูป และเมื่อทำการเขียนโปรแกรมโดยใช้หลักการ ของ VRML เข้าช่วย ก็จะได้ส่วนของ Hierarchy ที่สอดคล้องกันตั้งแต่ Root Node จนถึง Child node โดยการจะทำให้เกิดการเคลื่อนไหวนั้นก็เพียงแค่ เปลี่ยนค่าการ rotate ข้อมูลภายใน node นั้นๆ ซึ่งเป็นค่าภายในแกน X, Y หรือ Z เท่านั้น เพราะส่วนที่เป็น Child node จะเคลื่อนที่ตามส่วนที่เป็น Parent node ไปเอง ทำให้การเคลื่อนที่ดูสมจริงไล่ไปตามลำดับชั้นของโครงสร้าง

2.5.2 VRML และภาษาที่ใช้ในการสร้าง (Technology Promotion Associate. 2004)

VRMLย่อมาจากVirtual Reality Modeling Language (virtual Reality world) หรือเรียกว่า "เวอร์มอล" ซึ่งเป็นภาษาคอมพิวเตอร์ที่ใช้สร้างรูปเสมือนจริงเป็นรูปภาพกราฟิก3มิติประกอบกับความสามารถในการโต้ตอบกับผู้ใช้ทันทีที่ผ่านทางบราวเซอร์ (real-time interactive) ของระบบ world wide web (www) ของเครือข่ายอินเตอร์เนตเสมือนกับว่าผู้ใช้เข้าไปอยู่ในโลก 3 มิตินั้นจริงๆ นอกจากจะสามารถกราฟิก 3 มิติแล้วยังสามารถนำเสนอด้วยสื่อมัลติมีเดียเพื่อเพิ่มความสมจริงมากยิ่งขึ้น เช่นระบบเสียงที่เป็นลักษณะ 3 มิติ, เคลื่อนไหว ซึ่งสามารถโต้ตอบการเปลี่ยนแปลงมุมมองของผู้ใช้ได้ในเวลาจริง (real-time Interaction) โดยผ่านการรับรู้และเปลี่ยนแปลงมุมต่างๆ ภายในฉาก 3 มิติ ลักษณะเด่นๆ ของภาษา VRMLอาจจะสรุปได้ดังนี้

1. สร้างแบบจำลองกราฟิก 3 มิติ (3D graphic)
2. สร้างการโต้ตอบกับผู้ใช้ทันทีหรือที่เรียกว่าReal-time interactive
3. สร้าง แสง, เสียงในระบบ 3 มิติ (sound 3D)
4. สร้างภาพเคลื่อนไหว (animation)
5. มุมมองในการชมแบบจำลอง 3 มิติ 3 ทางคือ
6. การเดิน (walk)

7. การหมุน (rotate)

8. การบิน (fly)

ภาษา VRML อาศัยหลักการแสดงผลกราฟิกทั้ง 2 มิติและ 3 มิติ โดยอาศัยวิธีการแบบ "OpenGL"ซึ่งเป็นวิธีการแสดงผลของบริษัท Silicon graphic ที่ถูกนำเสนอในปี ค.ศ. 1992 ใช้สำหรับการค่าของวัตถุต่างๆ ไม่ว่าจะเป็นตำแหน่ง (coordinate) รายละเอียดพื้นผิว (texture) จุดเด่นของOpenGLคือไม่ขึ้นอยู่กับระบบปฏิบัติการจึงทำให้ผู้พัฒนาโปรมแกรมไม่จำเป็นต้องคำนึงถึงว่าจะใช้กับเครื่องชนิดใดและระบบปฏิบัติการแบบใด ในการสร้างภาพนั้น OpenGL จะทำการสร้างภาพวัตถุโดยสร้างรูปร่างพื้นฐานของวัตถุก่อน และเก็บใน Frame buffer ภาพวัตถุพื้นฐานในระบบคอมพิวเตอร์ คือจุด เส้นตรง ภาพหลายเหลี่ยมและบิตแมพ (bitmap) ภาพของวัตถุที่ถูกสร้างขึ้นนั้นส่วนประกอบแต่ละส่วนจะเป็นอิสระต่อกัน ดังนั้นในการเปลี่ยนแปลงลักษณะของวัตถุหรือภาพจะไม่ต้องกระทำทั้งวัตถุ เพียงแต่กระทำเฉพาะส่วนที่ต้องการเท่านั้น ไม่จำเป็นต้องเปลี่ยนแปลงทั้งวัตถุ เมื่อภาพของวัตถุที่แยกเป็นรูปพื้นฐานถูกเก็บใน Frame Buffer แล้วการเปลี่ยนแปลงใดๆ ในเรื่องของคุณสมบัติของวัตถุจะถูกควบคุมโดย openGL โดยตรงไม่ว่าจะเป็นการเปลี่ยนแปลงตำแหน่ง ขนาด สี หรือการกำหนดค่าความเข้มของแสงต่างๆ ส่วนการนำผลของค่าที่อยู่ใน Frame Buffer มาแสดงเป็นหน้าที่ของวินโดว์ ว่าจะแสดงค่าใดในจอภาพ

ภาษา VRML มีลักษณะเด่นในการแสดงวัตถุทั้งคงที่และเคลื่อนไหวและยังสามารถทำงานร่วมกับมัลติมีเดียอื่นๆ เช่น เสียง(voice), ภาพ(image), ภาพยนตร์(movies) โดยผ่านตัวประมวลผล คือ บราวเซอร์ (browser) นอกจากนั้นคือสนับสนุนลักษณะ 3 มิติ application programming interface (API) อีกด้วย ภาษา VRML ได้รับการรับรองจาก ISO (International Organization for Standardization) และ IEC (International Electronic Commission) ซึ่งเป็นหน่วยงานที่ทำทำหน้าที่ควบคุมมาตรฐานต่างของ internet โดยอนุญาตให้ใช้มาตรฐาน ISO/IEC 14772 ภายใต้หัวข้อของ Information Technology-Computer graphics and Image Processing-vertebral Reality Modeling Language (VRML)

ภาษา VRML ทำงานภายใต้แบบอย่างพื้นฐานของ web Browser-server ทั่วไป โดยอาศัยรูปแบบ URL ซึ่งเป็นรูปแบบมาตรฐานสำหรับระบบ WWW(world wild web) โดยกำหนดให้ขั้นต้นด้วย HTTP (Hypertext transfer Protocol) โดยเซิร์ฟเวอร์ภาษา VRML จึงถือว่าเป็นเครื่องมือในการสร้างโลกเสมือนจริงแบบใหม่ที่เรียกว่า "สังคมไร้พรมแดน"(cyberspace) และ "สังคมเสมือนจริง" (on-line Virtual Communications) ขึ้นมาจำลองสังคมมนุษย์ในโลกแห่งความจริงมาไว้ในโลกของคอมพิวเตอร์ 3 มิติ ทำหน้าที่แปลคำร้องของบราวเซอร์ ขณะดาวโหลด(download) Server ทำการส่งเอกสารที่เป็น Tag ของเอกสารหรือเรียกว่า Multipurpose Internet Mail Extensions (MIME) ซึ่ง

(MIME) ซึ่งภาษา VRML มีลักษณะเป็น cross-word/cross-vrml (x-world/x-vrml) โดยผู้ใช้สามารถดูด้วย บราวเซอร์ ที่เรียกว่า VRML browser ได้ โดยอาศัย ไฟล์ข้อมูลในรูปแบบ VRML(*.wrl) ซึ่งเป็นเป็นรูปแบบกลาง สำหรับแลกเปลี่ยนข้อมูล 3 มิติ อาศัยการนำเสนอวัตถุ(object) เป็นแบบเนสท์ (nested) โดยส่งข้อมูลรูปแบบทั้งหมดมาก่อนและตามด้วยระดับความละเอียดภายหลังอาศัยหลักการ LOD (level of detail) เปลี่ยนแปลงไปมาโดยอัตโนมัติและทำการเรนเดอร์ เพื่อสร้างแบบจำลองกราฟิก 3 มิติ ที่ VRML บราวเซอร์นั้นเอง ส่วนเอกสารที่เป็นเสียงหรือวีดีโอจะถูกส่งมาตามลำดับ

2.6 เครื่องมือที่ใช้ในการพัฒนา

การสร้าง Humanoid Model เพื่อนำไปใช้ในการทำ Real-time 3D tracking and Rendering นั้น จะแบ่งการทำงานออกเป็น 2 ส่วนด้วยกันคือ ในส่วนของการสร้างตัว Model สามมิติ และในส่วนของการ Programming โดยเครื่องมือหรือโปรแกรมที่ใช้ในการทำ ก็แตกต่างกันไปด้วย ดังนี้

2.6.1 ภาพกราฟิกและคุณลักษณะสำคัญ (Creative Master. 2002)

โดยปกติแล้ว ในทางเทคนิคจะแบ่งภาพออกเป็น 2 ชนิดคือ

- **ภาพกราฟิกแบบราสเตอร์ (Rastering Images)**

เป็นภาพที่เกิดจากการนำจุดสี่เหลี่ยมสีต่างๆ ที่เรียกกันว่าพิกเซล (Pixel) มาเรียงต่อกันจนเกิดเป็นรูปภาพ

ตัวอย่างที่เห็นได้ชัดเจนของภาพแบบราสเตอร์ก็คือ ภาพใน Wallpaper บนจอมอนิเตอร์ของเราเอง เมื่อลองใช้โปรแกรมขยายภาพขึ้นมาดู เราจะพบว่ามีพิกเซลสีต่างๆ เรียงตัวกันอยู่จนเกิดเป็นรูปภาพที่เห็นบนจอภาพนั่นเอง

- **ภาพกราฟิกแบบเวคเตอร์ (Vector Images)**

ภาพแบบนี้จะเกิดจากการนำสูตรทางคณิตศาสตร์มาใช้ในการสร้างเส้นสายต่างๆ ขึ้นมา โปรแกรมที่ถนัดในงานแบบเวคเตอร์มักจะเป็นโปรแกรมที่ใช้ในการวาดรูป เช่น Adobe Illustrator หรือ แม้แต่โปรแกรม 3 มิติ อย่างเช่น 3D Studio Max และ Maya ก็เป็นงานเวคเตอร์แบบหนึ่งเช่นเดียวกัน

ข้อดีของภาพแบบเวคเตอร์คือ ภาพจะถูกขยายขนาดได้มากๆ โดยที่ภาพจะไม่แตกเป็นจุดพิกเซล เหมือนภาพแบบราสเตอร์ แต่ก็มีข้อจำกัดในการใช้งานอยู่มากเหมือนกัน เรียกว่ามีความสามารถในการเขียนเส้นมากกว่าทำอย่างอื่น

2.6.2 ความสามารถของโปรแกรม 3 มิติ และงานที่เกี่ยวข้อง (Lead Animator. 2002)

งานด้าน Computer Graphic มีหลายหลากมากมาย งานส่วนใหญ่มักจะมีดังนี้คือ

- **งานภาพยนตร์และโทรทัศน์** เป็นวงการที่นำไปใช้งานมากที่สุด ไม่ว่าจะเป็นหนังที่ใช้ Computer Graphic ล้วนๆ หรือการใส่ เอฟเฟ็กต์ (Effect) ต่างๆ ลงไปในภาพยนตร์ ก็ตาม
- **วงการสถาปัตยกรรมและการก่อสร้าง** เช่น งานเขียนแบบอาคาร งาน Perspective หรืองานตกแต่งภายใน เป็นต้น
- **งานพัฒนาเกม** ถ้าลองสังเกตกันดูดีๆ จะเห็นว่าเกมในปัจจุบันกลายเป็นเกมแบบ 3 มิติกันไปหมดแล้ว เพราะฉะนั้นงานด้านนี้จึงมีความน่าสนใจไม่แพ้งานด้านอื่นๆ
- **งานด้านวิทยาศาสตร์** เช่น การสร้างโมเดลจำลองขึ้นมาเพื่อให้เห็นภาพของสิ่งต่างๆ จากการสมมติฐาน
- **งานด้านเว็บไซต์ (Website)** เป็นอีกสาขาหนึ่ง ที่งาน 3 มิติ กำลังมาแรงเช่นเดียวกัน และยังมีทางเลือกอื่นๆ เกิดขึ้นมากมาย ใครถนัดทางไหนก็นำไปพัฒนาต่อในทางนั้นๆ

2.6.3 ที่มาของโปรแกรม 3D Animation A/W Maya (Creative Master. 2002)

ในปี 1997 โปรแกรม Maya ออกมาสู่ตลาดเป็นครั้งแรกพร้อมความสามารถที่โดดเด่นในเรื่องการเคลื่อนไหวของตัวละครที่ละเอียดสมจริงมากที่สุดในขณะนั้น แต่ก็ยังมีบางส่วนที่มีข้อจำกัดอยู่บ้าง ทางผู้ผลิตโปรแกรมคือ Alias Wavefront ได้พัฒนาโปรแกรมอย่างต่อเนื่องจนมาถึงเวอร์ชันที่ 3 ที่ถือได้ว่ามีความนิยมอย่างมาก

จากการเพิ่มความสามารถในการเคลื่อนไหวแบบ Non-Linear เข้าไป พร้อมทั้งการสร้างลวดลายบนพื้นผิว 3 มิติด้วยการพ่น (Texture Paint) และ Plug-in เสริมอีกมากมาย ทำให้ Maya 3 ได้รับการต้อนรับจาก Hollywood เป็นอย่างดี พิสูจน์ได้จากการที่โปรแกรมถูกนำไปใช้ในภาพยนตร์หลายๆ เรื่อง จนถูกพัฒนาต่อมาเป็น Maya 5 ก็ยังคงความเป็นโปรแกรมที่มาแรง และมีความยืดหยุ่นในการทำงานได้ดีที่สุดเช่นเดียวกัน

สำหรับโปรแกรมที่ใช้ในการสร้าง Model มีหลายโปรแกรมด้วยกัน แต่ที่เลือกมาคือ Maya Version 4.5 และใช้ Plug in แปลงไฟล์ (File) .ma หรือ .dxf ที่ได้จากการทำด้วย Maya มาเป็น File format ในรูป .x สำหรับแสดงผลในโหมดภาพสามมิติของ DirectX 9.0 ในส่วนของตัว Model นั้นจะเป็น Model แบบพิเศษที่เรียนว่า Low Polygon Model เพื่อให้การแสดงผลรวดเร็ว และเป็น Real-time โดยคุณสมบัติของ Low Polygon Model คือ จะมีจำนวน Polygon น้อยมากๆ แต่ยังคงความสวยงามของตัว Model อยู่เหมือนเดิม การสร้างนั้น จึงไม่สามารถ นำ High Polygon 3D Model (ที่มี

ให้ Download Free ตาม Website ต่างๆ เช่น www.3dcafe.com) มาประยุกต์ใช้ได้ ต้องสร้างขึ้นมาเองใหม่ทั้งหมด แล้วจึงทำการ Map Texture เข้ากับตัว Model วิธีนี้จะเป็นวิธีที่ทำให้ Model ดูสวยงามและเป็น Low Polygon ด้วย

2.6.4 การสร้างโมเดลสามมิติจำลองแบบโครงสร้างร่างกายมนุษย์ ด้วยโปรแกรม Maya Version4.5 (Unlimited)

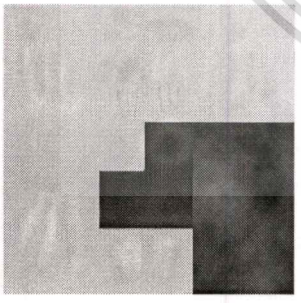
ก่อนอื่นต้องติดตั้งโปรแกรม แปลงไฟล์ .dxf หรือ .ma ให้เป็นไฟล์ .x ก่อน โดยโปรแกรมแปลงไฟล์ซึ่งเป็น plug in ของ Maya นั้นคือ

maya45and5xexport.zip

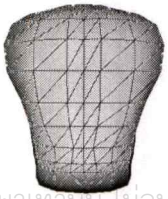
หลังติดตั้ง Plug in เรียบร้อยแล้ว จึงเริ่มใช้งานได้โดย การสร้างภาพสามมิติจำลองแบบโครงสร้างร่างกายมนุษย์ด้วยโปรแกรม Maya นั้น จะต้องทำการสร้างและ Save File เก็บไว้เป็นส่วนๆ โดยจะแยกตามส่วนของ Segment Node ที่มีการ define แบบ Hierarchy Structure ไว้ทั้งหมด 17 ส่วนดังแสดงในรูปที่2 โดยเริ่มที่ โหนดที่ 0 จนถึง โหนดที่ 16

หลักในการสร้าง Model รูปมนุษย์แบบ สามมิติเพื่อนำไปใช้กับโปรแกรมประยุกต์ C++ และ DirectX 9.0 นั้น ต้องพยายามสร้าง Model ให้ออกมา เป็นรูปแบบ low polygon [5] โดยจะแยกส่วนประกอบออกเป็นส่วนต่างๆ ตามที่กล่าวไว้แล้ว และทำการ map texture เข้าไปในภาพ โดยในที่นี้จะทำการ แบ่งการ Map ออกเป็น 3 ส่วนคือ ส่วนของ แขน/ขา (Arm/Legs) ศีรษะ (Head) และลำตัว (Body) ดังแสดงในรูปที่ 2.9

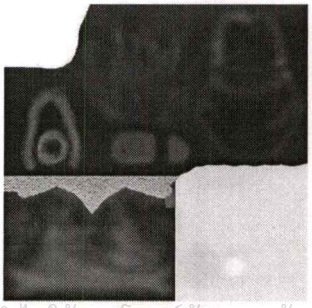
การ Map Texture

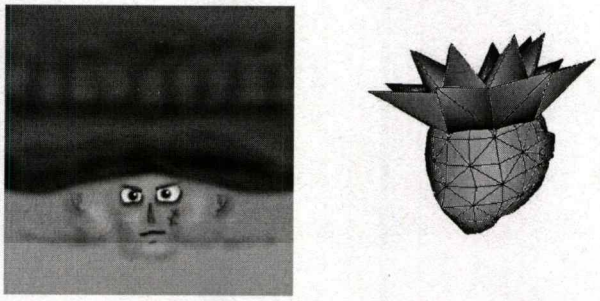


Arm/ leg



Body





Head

รูปที่ 2.9 แสดงการ Mapping Texture ในแต่ละส่วนของ Model

จากภาพ เป็นการทำการ Mapping texture โดยสร้าง texture จาก โปรแกรมสร้างภาพทั่วๆ ไป [6] เช่น Adobe Photoshop version 7.0 แล้วนำ Texture ที่ได้มาทำการห่อหุ้มเข้ากับตัว model ทำให้ได้สีสันสวยงาม โดยในที่นี้จะมีการ Mapping โดยใช้ Texture 3 ไฟล์ด้วยกัน โดยเทียบจากโทนสีและบริเวณที่อยู่คือ

1. ไฟล์ที่ใช้ทำส่วน Head (ส่วนศีรษะ) สีของ Texture ที่เกิดขึ้นจะเป็นสีของส่วนผม และใบหน้าเป็นส่วนใหญ่

2. ไฟล์ที่ใช้ทำส่วน Body (ส่วนลำตัว) Texture ส่วนใหญ่จะเป็นสีของเสื้อผ้า ซึ่งในที่นี้เป็นสีน้ำเงินเข้ม

3. ไฟล์ที่ใช้ทำส่วน Arm and Leg (ส่วนแขนและขา) สีที่ใช้เป็นสีเนื้อ

2.6.5 โปรแกรมที่ใช้ในการควบคุมการเคลื่อนไหว Model (MSDN. 2003)

เนื่องจากDirectX SDK นั้นสนับสนุนการทำงานด้วยภาษา VC++,VB และ C# ซึ่งมีผู้พัฒนาสร้างเครื่องมือสำหรับพัฒนาด้วย Delphi ด้วย แต่เหตุผลที่ผู้พัฒนาส่วนใหญ่แนะนำให้ทำงานDirectX SDKกับ VC++ คือDirectX ถูกออกแบบมาด้วยสถาปัตยกรรมCOM ซึ่งทำงานบน C++ เป็นหลักดั้งนั้นความเข้ากันได้จึงสูงกว่าภาษาอื่น ดังนั้นสำหรับโปรแกรมที่ใช้ในการ Coding เพื่อควบคุมการเคลื่อนไหวของตัว Model สามมิตินั้น จะใช้โปรแกรม Visual C++ พร้อมทั้งติดตั้ง DirectX 9.0 SDK เข้าไปและทำการเพิ่ม File ดังนี้

ในส่วนของ Include ทำการเพิ่ม File ข้างล่างนี้เข้าไป

DXSDK\INCLUDE

ในส่วนของ Library ก็ทำการเพิ่ม File เข้าไปเช่นกันดังนี้

DXSDK\LIB

**หมายเหตุ: สำหรับชุด (Set) ของภาพสามมิติที่ใช้ใน DirectX 9.0 นั้น เราจะเรียกชื่อใหม่ว่า Mesh

2.7 การเขียนภาษาเชิงวัตถุ และการสร้างโปรแกรมด้วย Microsoft Visual C++

การเขียนโปรแกรมแบบ Object Oriented Programming (OOP) เป็นแนวคิดในการเขียนโปรแกรมแบบหนึ่ง โดยจะไม่มุ่งเน้นไปในการเขียนโปรแกรมเพียงอย่างเดียว แต่เป็นการมองภาพรวมของสิ่งที่อยู่รอบๆ ตัวเราโดยสามารถให้คำจำกัดความได้ดังนี้ "วัตถุแต่ละอย่างต่างก็มีลักษณะและวิธีการใช้งานเป็นของตัวมันเอง" กล่าวคือ วัตถุแต่ละชนิดแต่ละอัน ต่างก็มีรูปร่างลักษณะ และการใช้งานที่แตกต่างกันออกไป เราจะเรียกคุณลักษณะของวัตถุนี้ว่า Attribute และเราจะเรียกวิธีการใช้งานวัตถุว่า Method ยกตัวอย่างเช่น จักรยานคันหนึ่งมีสีแดง สามารถใช้ขับขี่ได้ สีแดงจัดว่าเป็น Attribute ของจักรยาน ในขณะทีการใช้งานขับขี่ถือเป็น Method เป็นต้น

จะเห็นได้ว่าการเขียนโปรแกรมแบบ OOP เป็นแนวคิดที่คล้ายธรรมชาติของสิ่งของสิ่งหนึ่งซึ่งเราสามารถแบ่งแยกสิ่งต่างๆ ออกเป็นหลายๆประเภทได้ ถ้าเรานำเอาแนวคิดของ OOP มาใช้ในการเขียนโปรแกรม และการจัดการฐานข้อมูล จะพบว่า โปรแกรม หรือ ฟังก์ชัน จะมีความเป็นอิสระต่อกันอย่างเห็นได้ชัด อธิบายง่ายๆ ก็คือ โปรแกรม หรือฟังก์ชัน แต่ละตัวถึงแม้จะมาจากที่เดียวกัน แต่สามารถทำงานในคนละหน้าที่หรือเก็บข้อมูลคนละค่าได้

สำหรับคลาส (Class) ใน OOP คือกลุ่มของวัตถุที่คุณลักษณะพื้นฐาน และมีฟังก์ชันพื้นฐานที่เหมือนกัน เช่น คลาสคน คือ ไม่ว่าคนจะมีชื่อ อาชีพ เพศ อายุ หรือส่วนสูงเท่าใด ก็จะจัดอยู่ในคลาสของคนทั้งหมด ซึ่งก็จะมีคุณลักษณะ และฟังก์ชันพื้นฐานเหมือนกัน โดยผู้เขียนโปรแกรมไม่สามารถนำคลาสไปใช้งานได้โดยตรง ต้องมีการสร้าง Object (หรือที่เรียกว่า Instance) ขึ้นมาก่อน ซึ่ง Object ที่สร้างจาก Class ใดจะมีคุณลักษณะพื้นฐานมาจาก Class นั้นๆ ด้วย ดังนั้น การสร้างโปรแกรมด้วยวิธีการแบบ OOP จะทำให้การสร้างโปรแกรมมีความยืดหยุ่น และมีประสิทธิภาพสูงด้วย

โปรแกรมแบบ OOP มีคุณสมบัติพิเศษอีกอย่างหนึ่งที่ทำให้การเขียนโปรแกรมในแบบ OOP แตกต่างออกไป คือคุณสมบัติ การสืบทอด (Inheritance) การที่ OOP นำแนวคิดนี้มาใช้ ทำให้ผู้เขียนโปรแกรมสามารถนำเอา Source Code ที่อาจจะเป็นตัว Class ที่มีผู้เขียนไว้แล้ว สามารถนำกลับมาใช้ใหม่ได้อีก (Code Reuse) หรือมาเปลี่ยนแปลงแก้ไข ให้เหมาะสมกับงานเฉพาะอย่าง ทำให้การเขียนโปรแกรมเป็นไปได้อย่างรวดเร็วและเป็นการใช้ Source Code ที่มีอยู่ให้เกิดประโยชน์สูงสุด

โปรแกรมแบบ OOP นั้น ยังมีการกำหนดระดับการเข้าถึงข้อมูลภายใน Class เพื่อความปลอดภัย ในการเข้าถึงข้อมูลภายใน Class นั้นๆ ด้วย ซึ่งการกำหนดระดับการเข้าถึงข้อมูลภายใน Class นี้ เป็นคุณสมบัติอีกอย่างที่มีในแนวคิดแบบ OOP การเข้าถึงข้อมูลภายใน Class นี้ เป็นคุณสมบัติอีกอย่างที่มีในแนวคิดแบบ OOP การเข้าถึงข้อมูลภายใน (Member Variable) หรือ ฟังก์ชันภายใน (Member Function ที่เรียกว่า Method) แบ่งออกได้เป็น 3 ระดับดังต่อไปนี้คือ

- ระดับ Private เป็นระดับที่จะถูกกำหนดไว้โดยอัตโนมัติ โดยค่าที่ Set ไว้คือ Default ถ้าหากผู้เขียนโปรแกรม ไม่ได้ทำการกำหนดระดับเอาไว้ ระดับนี้เป็นระดับการป้องกันสูงสุด กล่าวคือ เป็นการป้องกันไม่ให้กระบวนการใดๆ ที่อยู่นอก Class เรียกใช้สิ่งที่อยู่ใน Class การจะเข้าถึงข้อมูลภายใน (Access Data) จะต้องผ่านจาก ฟังก์ชันที่เป็นของ Class นั้นอนุญาตให้ใช้ได้
- ระดับ Public เป็นระดับที่ตรงกันข้ามกับระดับ Private กล่าวคือ กระบวนการใดๆ ที่อยู่นอก Class นั้น สามารถเข้าถึงข้อมูลภายใน หรือฟังก์ชันภายในได้อย่างอิสระ ไม่จำเป็นต้องได้รับอนุญาตก่อนการเข้าใช้งาน
- ระดับ Protected คล้ายกับในระดับ Private แต่มีข้อแตกต่างกันคือ ในระดับ Protected นั้น Class ลูกที่สืบทอดคุณลักษณะจาก Class แม่ จะสามารถเข้าถึงข้อมูลภายใน Class แม่ได้โดยตรง

ความสามารถในการกำหนดระดับการเข้าถึงข้อมูลภายใน Class อาจเรียกว่า คุณสมบัติการห่อหุ้มวัตถุ หรือ Data Hiding

การเขียนโปรแกรมแบบ OOP นั้น ผู้เขียนโปรแกรม สามารถ ใช้คุณสมบัติของฟังก์ชันคอนสตรัคเตอร์ (Constructor) และ (Destructor) เพื่อเพิ่มความยืดหยุ่นให้กับโปรแกรมได้ ในการสร้าง Class แต่ละครั้งเราไม่จำเป็นต้องสร้าง คอนสตรัคเตอร์ และ ดีสตรัคเตอร์ เสมอไป การสร้างคอนสตรัคเตอร์และดีสตรัคเตอร์นั้น ขึ้นอยู่กับความจำเป็นในแต่ละสถานการณ์ และการดำเนินงานของ Class และใน Class หนึ่งสามารถมีคอนสตรัคเตอร์ได้หลายตัว โดยฟังก์ชันคอนสตรัคเตอร์ต้องมีชื่อฟังก์ชันเหมือนชื่อของ Class และไม่มีการคืนค่าใดๆ ออกมาเลย ฟังก์ชันคอนสตรัคเตอร์จะถูกเรียกโดยอัตโนมัติเมื่อมีการสร้าง Object และการสร้าง Object สามารถผ่าน Parameter (หรือไม่ผ่านก็ได้แต่จะเป็น Default) ซึ่ง Compiler จะเรียกฟังก์ชัน คอนสตรัคเตอร์ได้ตรงกัน คุณสมบัตินี้เรียกว่า สามารถโอเวอร์โหลดฟังก์ชันได้

ทั้งนี้ OOP ยังมีคุณสมบัติ Dynamic Binding (Late Binding) และ Polymorphism ซึ่งเป็นคุณสมบัติพิเศษที่ทำให้การเขียนโปรแกรมแบบ OOP แตกต่างจากการเขียนแบบโครงสร้างอีกด้วย

2.7.1 DirectX

DirectX เป็นชุดคำสั่งที่ใช้ในการควบคุมการเขียนโปรแกรม โดยจะเป็นชุดคำสั่งที่ทำให้ผู้เขียนโปรแกรมเรียกใช้ความสามารถด้าน Hardware ของระบบได้อย่างเต็มประสิทธิภาพ ไม่ว่าจะเป็นภาพ เสียง หรือการควบคุมเกม โดยจะช่วยอำนวยความสะดวกในการเขียนโปรแกรมเกี่ยวกับกราฟิกโดยตรง โดยจะ support บน ระบบปฏิบัติการ Windows ซึ่งเป็นระบบปฏิบัติการที่ปัจจุบันทั่วโลกมีผู้ใช้มากที่สุด

DirectX แบ่งออกเป็น 2 ส่วนคือ

1. DirectX Runtime Library

DirectX Runtime เป็นส่วนประกอบของ DirectX ที่ใช้ในการติดตั้งก่อนทำการ Run Program ที่เขียนขึ้นด้วยโปรแกรม DirectX

2. DirectX SDK (DirectX Software Development Kit)

DirectX SDK (DirectX Software Development Kit) เป็นชุดคำสั่งที่ออกแบบมาใช้ร่วมกันได้หลายภาษา เช่น C/C++ (Visual C++, Borland C++), Visual Basic หรือ Delphi และจะต้องทำการปรับให้ Software ต่างๆ ได้รู้จักกันด้วย

เนื่องจากมีรูปแบบคำสั่งที่ง่าย การเขียนโปรแกรมโดยใช้ DirectX ควบคุมจึงกำลังเป็นที่น่าจับตามองและเริ่มนิยมกันมากเพราะสามารถใช้ควบคุมการทำงานในรูปแบบกราฟิกโหมดได้ดี ในสัมมนา 1 ฉบับนี้จึงเลือกที่จะทำการ ศึกษาการโปรแกรมโดยใช้DirectX และนำมาพัฒนางานต่อไป โดยได้ทำการศึกษาการทำงานอย่างคร่าวๆ ของ DirectX ดังต่อไปนี้

DirectX ประกอบด้วย API ส่วนต่างๆ ดังนี้คือ

- **DirectDraw** เป็น API ใช้จัดการอุปกรณ์แสดงผล ควบคุมข้อมูลบิตแมป (Bitmap) หน่วยความจำที่ออกนอกพื้นที่สกรีน เป็นฟีเจอร์ (Feature) พื้นฐานที่ Direct3D สามารถทำได้ ซึ่งเป็นพื้นฐานสำคัญที่ใช้ในการสร้างแอพพลิเคชั่น สามมิติ
- **Direct3D** ในยุคแรกมี API อยู่ 2 โหมด คือ Immediate (IM) การ Render Object ทำได้โดยฉับพลันและตรงตามความต้องการของโปรแกรมเมอร์ เป็นโหมดที่ใช้งานยากแต่ก็มีความยืดหยุ่นสูง แสดงผลได้รวดเร็ว และมีประสิทธิภาพ อีกโหมดหนึ่งก็คือ Retained (RM) API เก็บซีน (Scene) ลงในฐานข้อมูล แล้วนำมา Render ทั้งหมดในคราวเดียวกัน เป็นโหมดที่สร้างมาเป็นเลเยอร์(Layer) บนสุดของโหมด IM แต่ไม่ค่อยมีความยืดหยุ่น รวมทั้งไม่สนับสนุนเทคโนโลยี

ใหม่ๆเช่น Multitexturing, Bump Mapping, Hardware Transformation และ Lighting ดังนั้น ความสามารถทั้งหมดของโปรแกรมสามมิติ จึงควรเขียนขึ้นมาด้วยการใช้โหมด IM

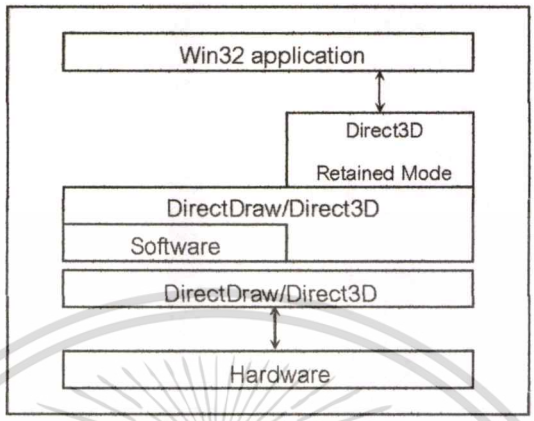
- **DirectMusic** เป็นชุด API ที่ทำงานร่วมกับข้อมูลประเภท Message-based Musical Data ซึ่งเป็นข้อมูลที่แปลงมาจาก Wave Sample โดยเป็นไปตามมาตรฐานของ DLS (Down-loadable Sound) สามารถสร้างเพลงได้ตามที่เรากำหนดไว้ด้วย
- **DirectSound** เป็นชุดเครื่องมือ API ที่ใช้ในการจัดการเสียงแบบสเตอริโอและแบบสามมิติอย่างมีประสิทธิภาพ การทำซิมูเลชั่น(Simulation)เสียงแบบสามมิติ ทำให้แอพพลิเคชั่นให้เสียงได้อย่างสมจริงสมจัง การได้ยินเสียงจากทางซ้าย ทางขวา หรือด้านบนเป็นต้น
- **DirectPlay** สนับสนุนการทำงานผ่านเครือข่าย โมเด็ม และสนับสนุนการออกแบบเกมต่างๆ แบบออนไลน์ด้วย
- **DirectInput** เป็น API ของ DirectX ที่สนับสนุนความเร็วต่ำ(Low-Latency Input) ซึ่งเป็นอุปกรณ์อินพุตส่วนใหญ่ที่มีในปัจจุบัน ใน DirectX 7 ขึ้นไป ได้เพิ่มการสนับสนุนอุปกรณ์อินพุตทุกชนิดที่มีความสามารถแสดงปฏิกิริยาย้อนกลับ เช่น จอยสติก หรือพวงมาลัยแบบสั่นได้ อุปกรณ์เหล่านี้สามารถจำลองสถานการณ์บางส่วนให้สมจริง เช่น การจำลองสถานการณ์เมื่อขับรถชนบนถนน จะเกิดแรงสั่นสะเทือน อย่างแรง เป็นต้น
- **DirectSetup** ใน API ส่วนนี้ จะมีหน้าที่ติดตั้งคอมโพเนนต์ DirectX Runtime ให้กับระบบโดยอัตโนมัติ ถ้าหากระบบยังไม่ได้ติดตั้ง โดยไมโครซอฟต์อนุญาตให้ใช้ได้โดยไม่เสียค่าใช้จ่ายใดๆ ดังนั้นจึงไม่ต้องมากังวลกับเรื่องของค่าลิขสิทธิ์ ทำให้สามารถติดตั้งกี่ครั้งก็ได้

ในส่วนที่จะทำการศึกษานั้นคือส่วนของ Direct3D ซึ่งเกี่ยวข้องโดยตรงกับการทำภาพสามมิติซึ่งมีการทำงานดังนี้

Direct3D มีขึ้นครั้งแรกใน DirectX 2.0 ประมาณช่วงปี ค.ศ. 1996 หลังจากนั้นภายในระยะเวลาอันสั้น Direct3D ได้กลายเป็นชุด API สำหรับงานสามมิติที่ได้รับความนิยมอย่างสูงในท้องตลาด ที่การทำเกมและซิมูเลชันส่วนใหญ่เลือกใช้ รวมทั้งได้รับการสนับสนุนจากผลิตภัณฑ์เร่งความเร็วสามมิติ ปัจจุบันฮาร์ดแวร์เร่งความเร็วกราฟิกพัฒนาไปจนสามารถแสดงโพลีกอน (Polygon)ได้หลายล้านโพลีกอนต่อวินาที ฮาร์ดแวร์เร่งความเร็วที่นำมาเชื่อมการทำงานกับ Direct3D มีความเร็วที่สูงมากและลดต้นทุนของแอพลิเคชันสามมิติ เราจึงไม่จำเป็นต้องลงทุนสูงมากนักเพื่อแลกกับประสิทธิภาพการทำงานที่สูงขึ้น

การทำงานร่วมกับฮาร์ดแวร์เร่งความเร็วกราฟิกของแอพพลิเคชัน Direct3D ทั้งโหมด RM และโหมด IM ทำงานคล้ายคลึงกัน โดย Direct3D จะติดต่อฮาร์ดแวร์ผ่านทาง HAL(Hardware Abstraction Layer) แต่ถ้าไม่มี Direct3D ก็จะติดต่อผ่านทาง HEL(Hardware Emulation Layer)

แทน โดยในรูปที่ 2.10 นี้จะแสดงให้เห็นว่า Direct3D เป็นเหมือนอินเตอร์เฟซที่จะใช้เชื่อมต่อไปยัง DirectDraw โดยตรง

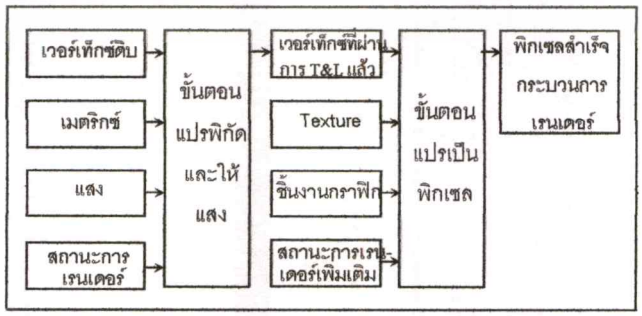


รูปที่ 2.10 แสดงความสัมพันธ์ที่ Direct3D มีต่อสภาพแวดล้อมของ DirectX และระบบ

2.8 หลักการของกระบวนการสร้างภาพ 3 มิติ

การสร้างภาพสามมิติ กระบวนการแปรพิกัดและให้แสง

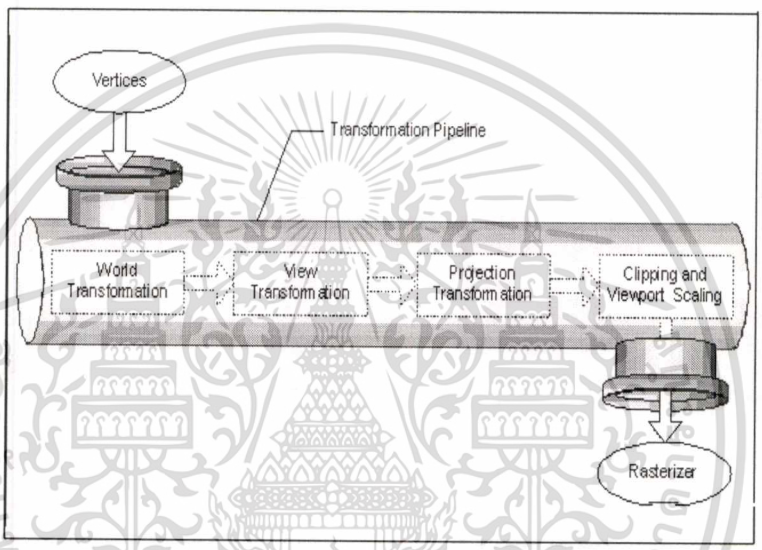
งานในกระบวนการเรนเดอร์อ็อบเจ็กต์สามมิติแบ่งเป็นสองขั้นตอน ขั้นตอนแรกคือกระบวนการการแปรพิกัดและให้แสงหรือที่เรียกย่อว่า T&L Pipeline หรือ Transformation and Lighting นั่นเอง ขั้นตอนนี้เป็นกระบวนการแปรพิกัดของเวอร์เท็กซ์จากพิกัดที่มีรูปแบบทศนิยมไปสู่พิกัดแบบพิกเซลตามมุมมองของกล้องจำลองภายในซีน รวมทั้งยังนำไปใช้ในการทำเอฟเฟ็กต์ของแสงและกระบวนการขริบและปรับขนาดของวิว ขั้นตอนที่สองคือกระบวนการแปรมาสู่พิกเซล (Rasterization) เป็นขั้นตอนลงจุด เส้น และรูปสามเหลี่ยมด้วยชิ้นงานกราฟิกที่ผ่านกระบวนการ T&L ออกมาแล้ว รูปร่างชิ้นงานกราฟิกสุดท้ายที่ผ่านกระบวนการแปรเป็นพิกเซลแล้วจะถูกวาดลงพื้นผิวของ DirectDraw โดยแสดงงานในแต่ละขั้นตอนการเรนเดอร์ชิ้นงานกราฟิก สามมิติได้ดังรูปที่ 2.11



รูปที่ 2.11 แสดงขั้นตอนทั้งสองในกระบวนการเรนเดอร์อ็อบเจ็กต์สามมิติ

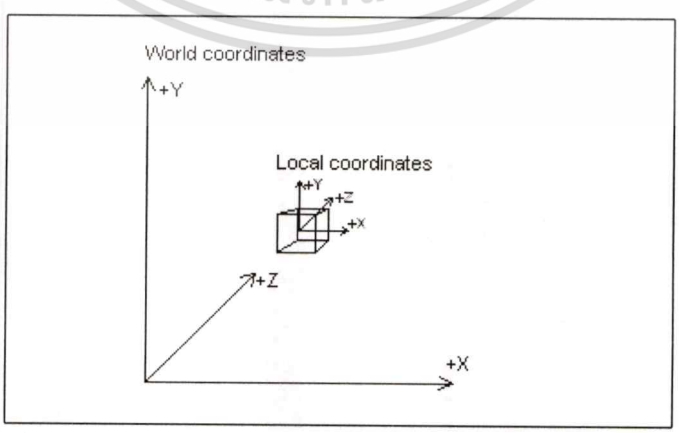
ภาพรวมของเวอร์เท็กซ์กับกระบวนการแปรพิกัดและให้แสง (T&L Pipeline)

การทำงานเริ่มต้นโดยการส่งเวอร์เท็กซ์ดิบเข้าสู่สายการผลิตทางด้านหนึ่ง จากนั้นเวอร์เท็กซ์ดิบดังกล่าว ก็จะผ่านกระบวนการออกมาที่ปลายอีกด้านหนึ่ง กระบวนการทั้งหมดนี้คือ T&L Pipeline เมื่อเวอร์เท็กซ์เข้าสู่สายงาน มันจะถูกแปรพิกัด ให้แสง ขริบขอบที่ไม่แสดงผล วางจุดหมายอ้างอิงลงพื้นที่สกรีน และปรับขนาดมุมมองตามวิวพอร์ต เสร็จสิ้นแล้วเวอร์เท็กซ์ที่สำเร็จจากกระบวนการดังกล่าวก็พร้อมที่จะส่งมอบให้กับกระบวนการแปรเป็นพิกเซลต่อไป ดังรูปที่ 2.12



รูปที่ 2.12 แสดงกระบวนการ T&L Pipeline

การแปรไปสู่พิกัดเวิลด์



รูปที่ 2.13 แสดงภาพพิกัดเวิลด์ (World Coordinates)

ในระหว่างกระบวนการแปรพิกัดและให้แสง พิกัดของอ็อบเจ็กต์ทั้งหมดที่นำมาเรนเดอร์ต้องแปรให้มาอยู่ในระบบพิกัดเดียวกันนั่นคือ พิกัดเวิลด์(World Transformation)ดังรูปที่ 2.13 โดยขั้นตอนแรกในกระบวนการ T&L Pipeline คือใช้เมตริกซ์เวิลด์ ซึ่งเป็นเมตริกซ์ที่ใช้สำหรับแปรพิกัดตำแหน่งของเวอร์เท็กซ์ในอ็อบเจ็กต์ จากพิกัดท้องถิ่นไปสู่พิกัดเวิลด์ และยังนำมาใช้ในการหมุน (Rotation) อ็อบเจ็กต์รอบแกน X,Y หรือ Z (โดยดูภาพแสดงระบบแกน สามมิติได้จากรูปที่ 2.11) การเคลื่อนย้าย (Translation) ตามแกน X, Y หรือ Z และการขยายขนาด (Scaling) อ็อบเจ็กต์ให้ใหญ่ขึ้นหรือเล็กลง ร่วมกันอีกด้วย โดยจะมีรูปแบบการใช้งานเมตริกซ์ดังต่อไปนี้

การเคลื่อนย้าย (Translation)

จากตำแหน่ง (x, y, z) ไปที่ (x', y', z')

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

สมการคือ

$P' = T \cdot P$ โดย T คือตำแหน่งที่จะย้ายวัตถุไป

การหมุน (Rotation)

จากตำแหน่ง (x, y, z) ไปที่ (x', y', z') ในแกน z

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

สมการคือ

$P' = R_Z(\theta) \cdot P$ โดย R คือตำแหน่งที่จะหมุนวัตถุในแกน z

จากตำแหน่ง (x, y, z) ไปที่ (x', y', z') ในแกน y

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 0 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

สมการคือ

$P' = R_Y(\theta) \cdot P$ โดย R คือตำแหน่งที่จะหมุนวัตถุในแกน y

จากตำแหน่ง (x, y, z) ไปที่ (x', y', z') ในแกน x

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

สมการคือ

$P' = R_X(\theta) \cdot P$ โดย R คือตำแหน่งที่จะหมุนวัตถุในแกน x

การปรับขนาด (Scaling)

จากขนาดเดิม (x, y, z) ไปเป็น (x', y', z')

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

สมการคือ

$P' = S \cdot P$ โดย S คือขนาดของวัตถุ

การแปรไปสู่พิกัดวิว

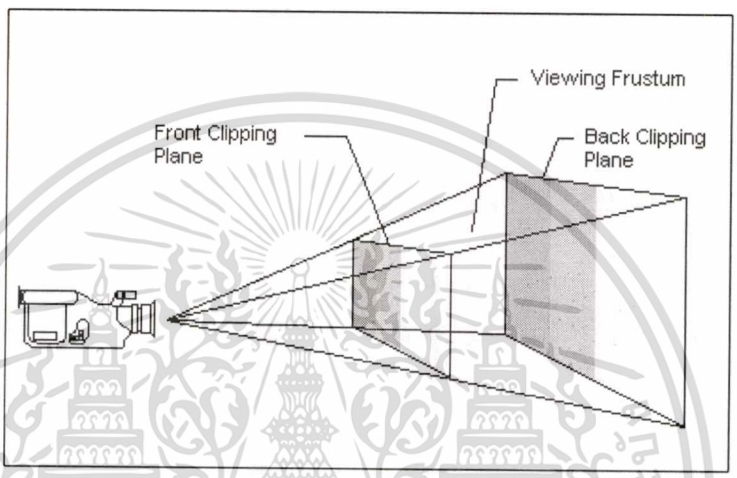
งานขั้นที่ 2 ของขั้นตอน T&L Pipeline เป็นการแปร พิกัดเวิลด์ ไปสู่พิกัดที่มองออกมาจากกล้องหรือพิกัดมุมกล้อง กล้องที่ว่านี้ก็คือกล้องจำลองซึ่งตั้งอยู่ ณ ตำแหน่งจุดกำเนิด และสามารถกำหนดให้อยู่ตามแนวแกน z เรื่อยลงมาตามแนวแกนด้านบวกได้ นอกจากนี้ในขั้นตอนการให้แสงทั้งหมด ก็จะถูกแปรเข้ามาสู่ระบบพิกัดมุมมองกล้องด้วย

การให้แสง

ขั้นตอนนี้ เอฟเฟ็กต์ของแสงแต่ละแบบจะถูกคำนวณผลที่มีต่อเวอร์เท็กซ์ทั้งหมด โดยการให้แสงจะพิจารณาจากเวกเตอร์ Normal (เวกเตอร์ตั้งฉากที่ออกมาจากเวอร์เท็กซ์ในโพลีกอน) สี และคุณสมบัติของวัสดุที่แปะอยู่บนเวอร์เท็กซ์ ทั้งหมดนี้ใช้สำหรับคำนวณผลของเอฟเฟกต์ที่มีต่อเวอร์เท็กซ์ หลังจากนั้น งานนี้จะเสร็จสิ้นเมื่อจัดเก็บสีของเวอร์เท็กซ์แต่ละตัวลงบนโครงสร้างข้อมูลของตัวเอง

การแปรไปสู่พิกัดโปรเจ็กต์ชันและ Viewing Frustum

กระบวนการนี้ทำให้อ็อบเจ็กต์ปรากฏตามความลึกภายในซีน(Scene) โดยทำให้อ็อบเจ็กต์ที่อยู่ห่างไกลมีขนาดเล็กกว่าอ็อบเจ็กต์ที่อยู่ใกล้จุดมองมากกว่า หลังจากเสร็จสิ้นกระบวนการนี้แล้ว เวอร์เท็กซ์ทั้งหมดจะถูกแปรไปอยู่ในพิกัดโปรเจ็กต์ชัน ส่วน Viewing Frustum คือการตัดมุมมองของกล้องให้อยู่ในระหว่างพิกัดแกน Z ด้านหน้าและหลัง ดังรูปที่ 2.14



รูปที่ 2.14 แสดง Viewing Frustum

การคลิบ (Clipping)

เป็นขั้นตอนที่ทำเพื่อให้มั่นใจว่าอ็อบเจ็กต์ที่อยู่นอกขอบเขตของ Viewing Frustum จะไม่ถูกนำมาเรนเดอร์ และอ็อบเจ็กต์ทั้งหมดที่อยู่ภายใน Viewing Frustum จะถูกวาดขึ้นโดยไม่มีพิกเซลใดเล็ดลอดออกจากขอบเขตการมอง

การปรับขนาดวิวพอร์ต (View Port)

เป็นงานสุดท้ายในขั้นตอน T&L Pipeline เป็นงานปรับเวอร์เท็กซ์ให้มีขนาดเหมาะสมกับวิวพอร์ต (ขอบเขตของการมอง) ที่เราต้องการ โดยวิวพอร์ตอนุญาตให้เราสามารถกำหนดวิธีที่ใช้ในการแมป (Map) ภาพอิมเมจ (Images) ลงบนพื้นที่เป้าหมายได้ โดยสามารถกำหนดให้มีทั้งการโยกย้ายและปรับขนาดด้วย

บทที่ 3

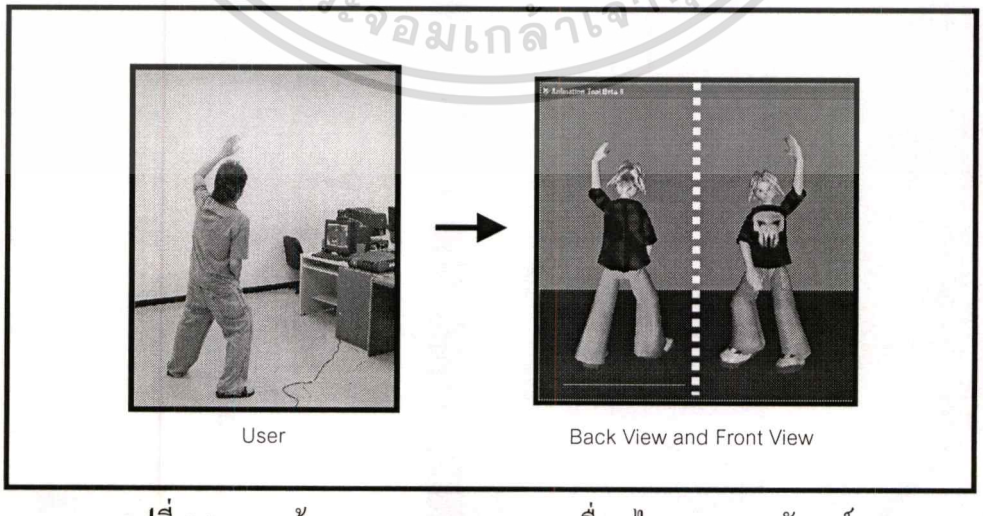
การออกแบบ การทดลอง และการพัฒนาระบบงาน

ในบทนี้จะเป็นการนำหลักการทั้งหมดรวมทั้งทฤษฎีที่ได้กล่าวไว้จากบทที่ผ่านมา มาทำการสร้างโปรแกรมประยุกต์โดยมีขั้นตอนทั้งหมดดังต่อไปนี้

3.1 การออกแบบระบบ

สำหรับตัวโครงงานนี้ทั้งหมด จะประกอบด้วยส่วนย่อยๆ ทั้งหมด 4 ส่วนด้วยกันโดยส่วนที่นำเสนอนั้น เป็นส่วนสุดท้ายของงาน คือ การแสดงผลลัพธ์ (Output) จากการรับค่าการเคลื่อนไหวในระบบสามมิติแนวแกน X,Y,Z มาแสดงผลโดยผ่านทางโมเดลที่ได้ทำขึ้นไว้แล้ว เนื่องด้วยการทำโครงงานนี้ ประกอบด้วยหลายส่วน ทำให้ต้องใช้การเขียนโปรแกรมช่วยในการรับ Input ค่าข้อมูลเข้ามาเพื่อทำการทดลอง

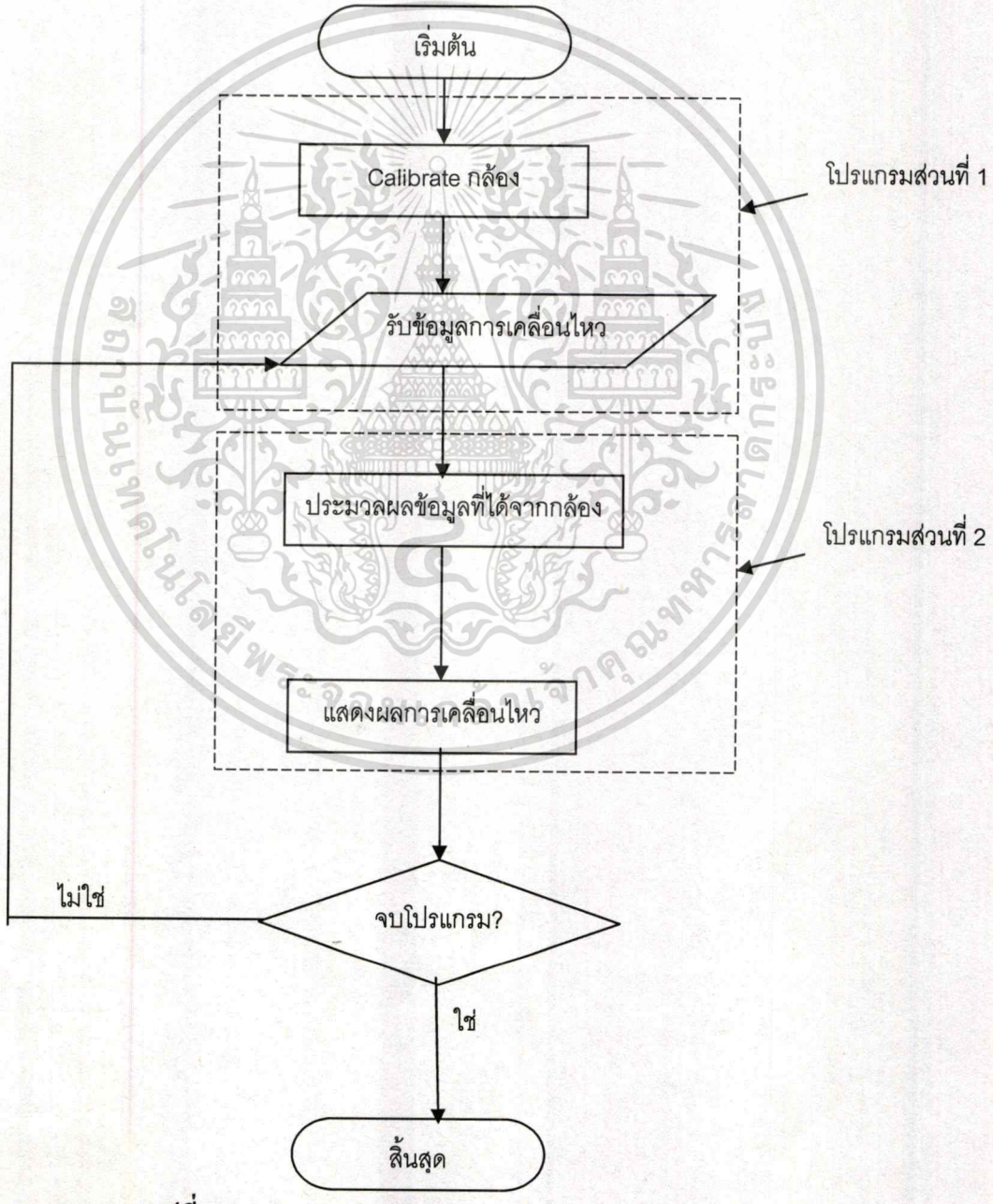
เมื่อระบบทั้งหมดเสร็จสมบูรณ์แล้วก็จะทำการรวมส่วนต่างๆเข้าด้วยกัน ทำให้สามารถรับค่าจากกล้องแล้วนำมาประมวลผลเพื่อทำการแสดงผลออกมาเป็นการเคลื่อนไหวของตัวโมเดลสามมิติได้ ซึ่งถ้าการรับข้อมูลจากกล้องมาไม่เกิดการผิดพลาด ก็จะทำให้การเคลื่อนไหวของโมเดลเสมือนจริงดังผู้ที่เข้าไปทดสอบเคลื่อนไหวร่างกายด้วย โดยเมื่อระบบทั้งหมดเสร็จสมบูรณ์ ก็จะได้ผลลัพธ์ดังรูปที่ 3.1



รูปที่ 3.1 แสดงผู้ทดสอบ และ ภาพการเคลื่อนไหวตามของตัวการ์ตูน

Flowchart แสดงการทำงานของโปรแกรมทั้งระบบ

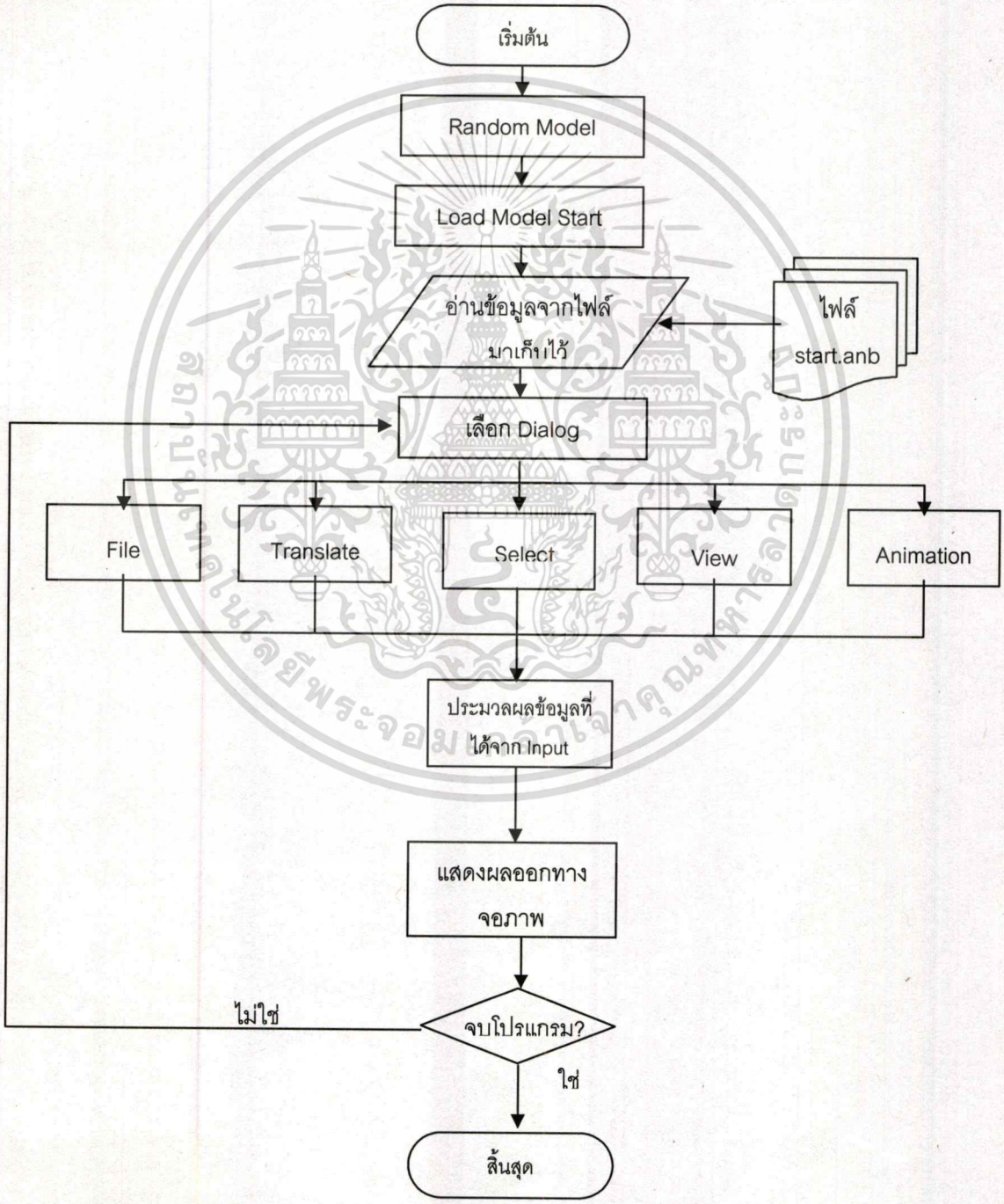
การทำงานของระบบทั้งหมดคือ ระบบ Motion Capture เมื่อนำมารวมกันแล้ว คือระบบจะรับค่าข้อมูลการเคลื่อนไหวจากผู้ใช้งานแล้วนำมาทำการ Calibrate และประมวลผล เป็นส่วนแรกของระบบคือส่วนที่ 1 เรียกว่า Tracking ซึ่งมีผู้พัฒนาคือ คุณสน หาญวงศ์ แล้วจึงส่งข้อมูลต่อให้กับโปรแกรมประยุกต์รับค่าเพื่อแสดงการเคลื่อนไหวของผู้ใช้ออกมาเป็นภาพสามมิติของตัวการ์ตูนแบบ Real-Time คือส่วนที่ 2 เรียกว่า Rendering ดังรูปที่ 3.2



รูปที่ 3.2 แสดงแผนการทำงานของระบบ Motion Capture

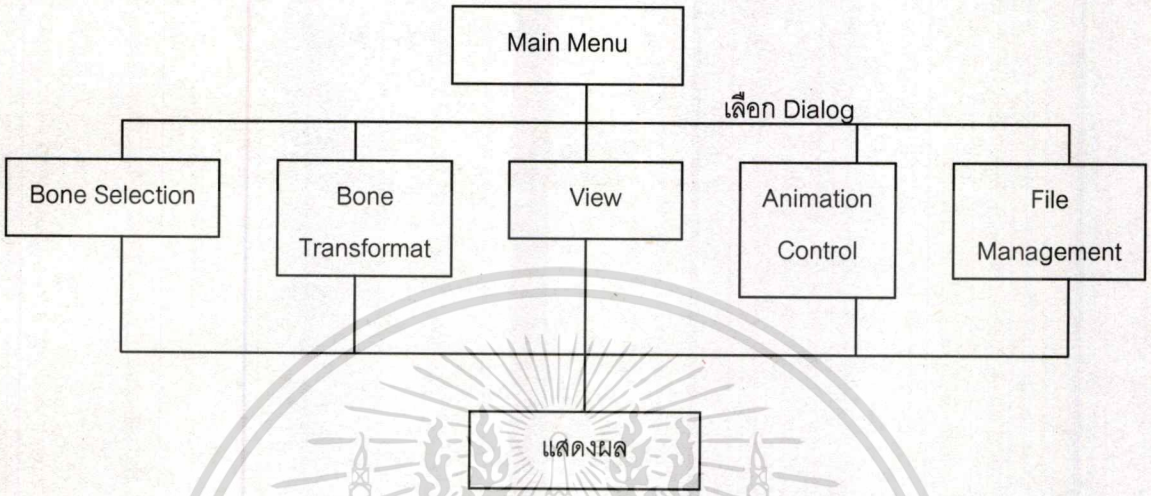
สำหรับในส่วนงานของโครงการนี้ เนื่องด้วยระบบรวมทั้งหมดยังไม่เสร็จสมบูรณ์ จึงมีการแยกระบบออกเป็นสองส่วน โดยส่วนที่ได้นำเสนอในโครงงานนี้คือ ส่วนของการ Rendering โมเดล โดยใช้ข้อมูลสมมุติที่ได้สร้างขึ้นจาก Input Dialog เพื่อสร้างชุดข้อมูลให้เกิดการเคลื่อนไหวของโมเดล โดยมีขั้นตอนการทำงานดังต่อไปนี้

Flowchart แสดงการทำงานของโปรแกรมประยุกต์ควบคุมโมเดลสามมิติ



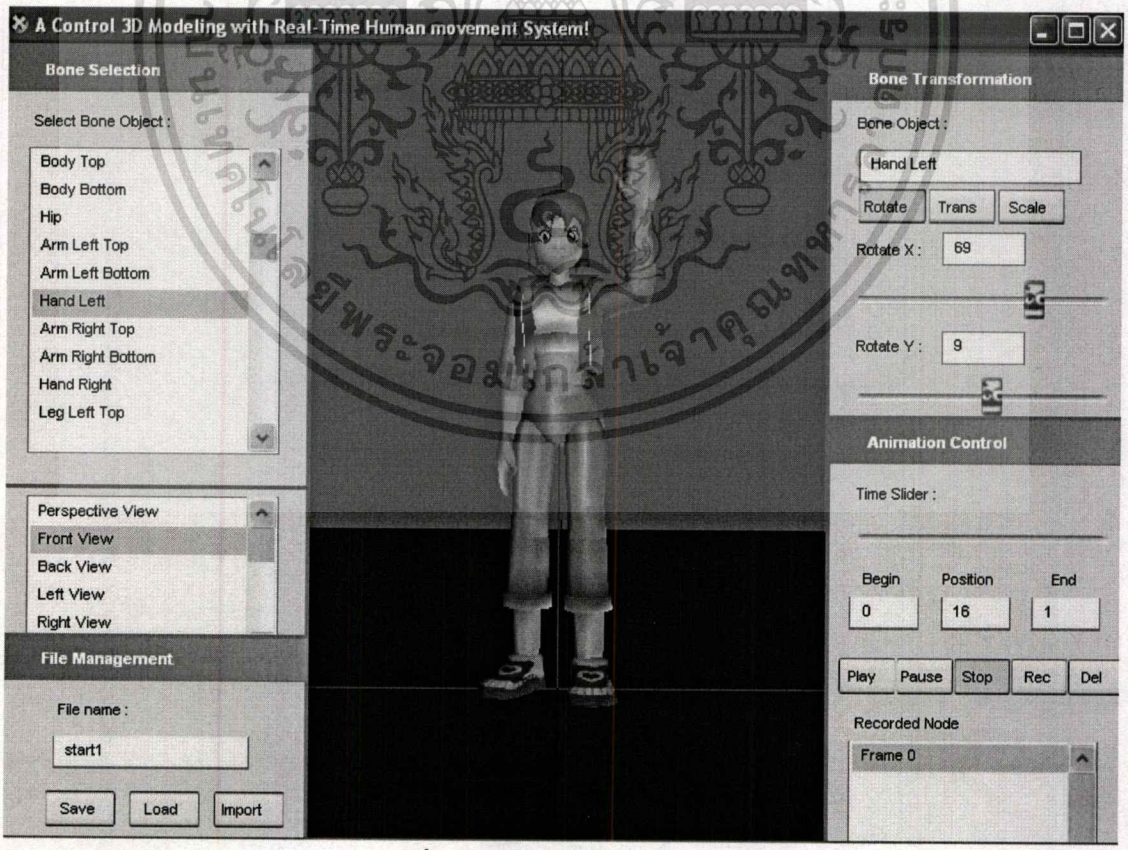
รูปที่ 3.3 แสดงแผนการทำงานของระบบย่อย

สถาปัตยกรรมของระบบและโมดูลต่างๆ



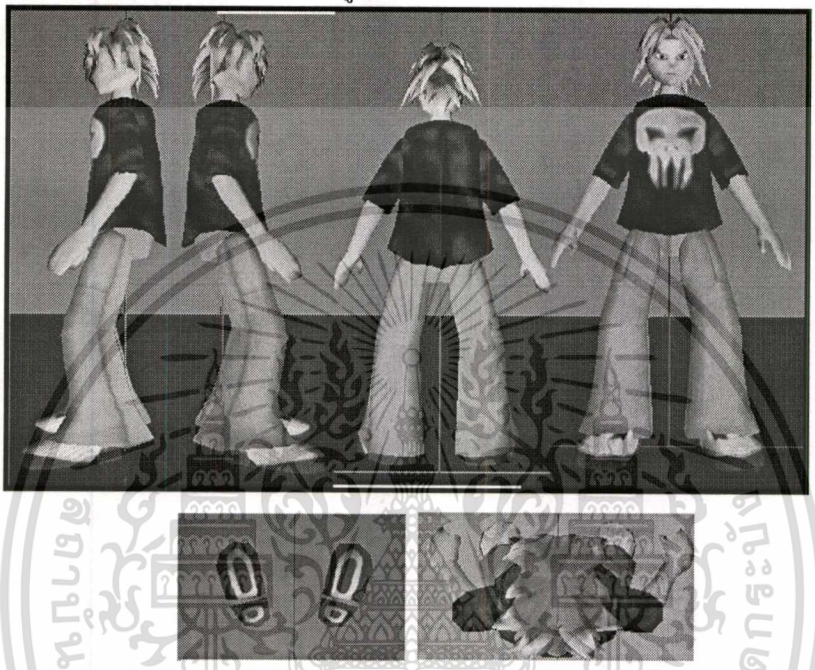
รูปที่ 3.4 แสดงสถาปัตยกรรมของระบบ

หน้าจอโดยรวม



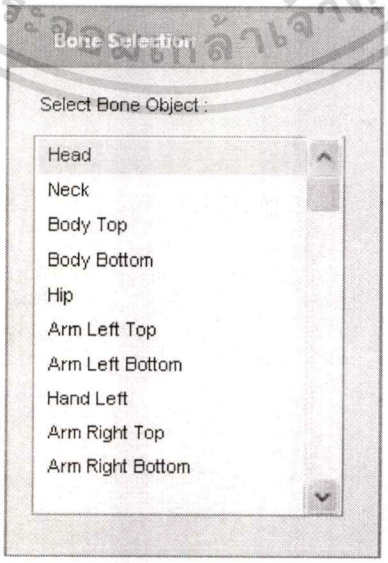
รูปที่ 3.5 แสดงหน้าจอโดยรวมของระบบ

ทั้งนี้เรายังสามารถเพิ่ม Feature ต่างๆ ให้กับระบบได้อีกเช่น การใส่ฉากหลังเข้าไปหรือการที่ผู้ทดสอบ สามารถเลือกฉากหลังระหว่างการเล่นได้ การเปลี่ยนตัวโมเดลที่ใช้ทดสอบ การเปลี่ยนมุมมองต่างๆ ในการดู เช่น Perspective View, Front View, Back View, Left View, Right View, Top View หรือ Bottom View เป็นต้น ดังรูปที่ 3.6



รูปที่ 3.6 แสดงมุมมองต่างๆ ของโปรแกรม

ในส่วนของการทำงาน เมื่อเปิดโปรแกรม ระบบก็จะเริ่มจากการ Load ตัวโมเดลขึ้นมา โดยจะมีหน้าต่างที่เป็นไดอะล็อก (Dialog) สำหรับเลือกการ Input ค่าให้โมเดลเคลื่อนไหวขึ้นมา 5 ไดอะล็อก ดังนี้



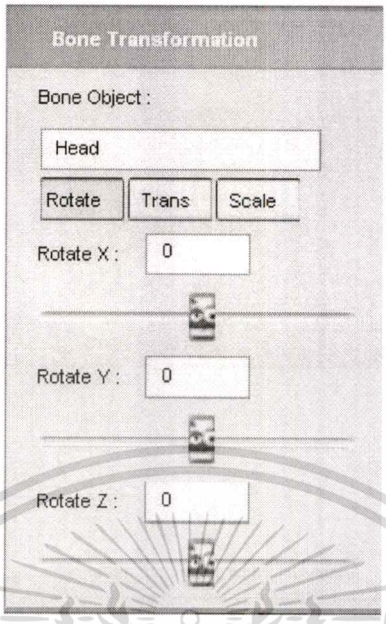
รูปที่ 3.7 แสดงภาพ Dialog box Bone Selection

จากรูปที่ 3.7 สำหรับเลือกส่วนที่จะ input ข้อมูลในโปรแกรม ในการทำการป้อนข้อมูลเข้าไปให้โมเดลแสดงผลการเคลื่อนไหวนั้น เป็นการเปลี่ยนข้อมูลในโหนดนั้นๆ ให้มีค่าเปลี่ยนแปลงไป โดยสามารถเลือกส่วนที่ต้องการให้เคลื่อนไหวได้ทั้งหมด 17 ส่วนคือ

ตารางที่ 3.1 แสดงส่วนต่างๆ ของโมเดลสามมิติและรายละเอียด (Lam. 1997)

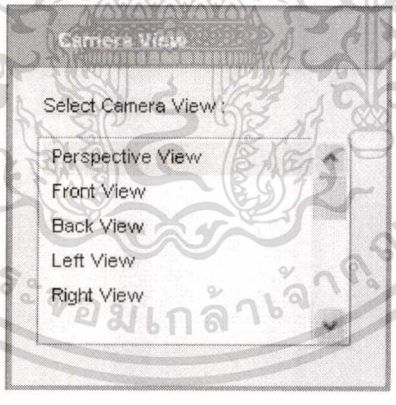
Select bone object:	Description
Head	สำหรับส่วนของศีรษะ
Neck	สำหรับส่วนของคอ
Body Top	สำหรับส่วนของลำตัวท่อนบน
Body Bottom	สำหรับส่วนของลำตัวท่อนล่าง
Hip	สำหรับสะโพก
Arm Left Top	สำหรับแขนซ้ายท่อนบน
Arm Left Bottom	สำหรับแขนซ้ายท่อนล่าง
Hand Left	สำหรับมือซ้าย
Leg Left Top	สำหรับขาซ้ายท่อนบน
Leg Left Bottom	สำหรับขาซ้ายท่อนล่าง
Foot Left	สำหรับเท้าข้างซ้าย
Arm Right Top	สำหรับแขนขวาท่อนบน
Arm Right Bottom	สำหรับแขนขวาท่อนล่าง
Hand Right	สำหรับมือขวา
Leg Right Top	สำหรับขาขวาท่อนบน
Leg Right Bottom	สำหรับขาขวาท่อนล่าง
Foot Right	สำหรับเท้าขวา

หลังจากเลือกส่วนที่ต้องการให้เคลื่อนไหวแล้ว ให้เลือก Dialog Bone Transformation สำหรับทำการเปลี่ยนค่าต่างๆ ในโหนดทั้ง 17 โหนด โดยจะสามารถเปลี่ยนค่าการ Translate Rotate และ scaling ได้ และยังสามารถเลือกแกนที่ต้องการจะให้เกิดการเปลี่ยนแปลงค่าได้อีกด้วยคือ แกน X, Y, Z



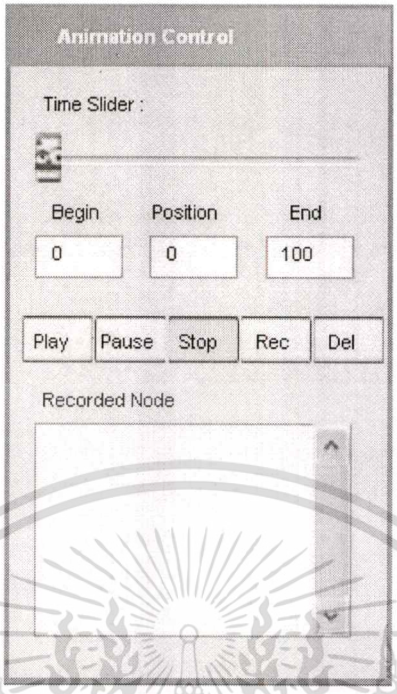
รูปที่ 3.8 แสดงภาพ Dialog box Bone Transformation

เมื่อทำการเปลี่ยนค่าใน Dialog สำหรับ input ข้อมูลในส่วนต่างๆของโมเดล ตามรูปที่ 3.8 จะทำให้เกิดการเคลื่อนไหวของโมเดลในส่วนที่ถูกเลือกเอาไว้



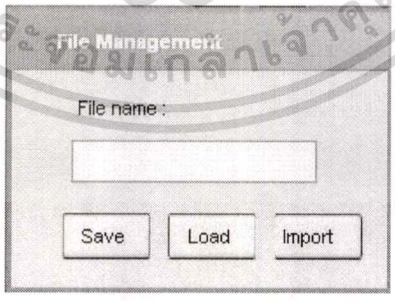
รูปที่ 3.9 แสดงภาพ Dialog box Camera View

สำหรับในรูปที่ 3.9 เป็น Dialog box Camera View สำหรับ เลือกมุมมองต่างๆ ในการดูโมเดลสามมิติเคลื่อนไหว แล้วแต่ความชอบหรือความถนัดในการมองของแต่ละบุคคลที่ใช้ตัวโปรแกรม โดยจะมีคีย์ (Key) ลัดในการดูมุมมองจากแป้นพิมพ์คือ F1-F7 ไล่ลำดับไปตาม Dialog จากบนลงล่าง เช่น ปุ่ม F1 บน Keyboard คือ ปุ่มสำหรับมุมมองแบบ Perspective View ปุ่ม F2 สำหรับมอง Front View เป็นต้น



รูปที่ 3.10 แสดงภาพ Dialog box Animation Control

ในรูปที่ 3.10 เป็น Dialog box Animation Control สำหรับบันทึกข้อมูลก่อนนำไปแสดงผลหรือเก็บลงไฟล์ โดยจะบันทึกเฟรม (Frame) ได้ตามจำนวนที่เรากำหนด สามารถทำการกำหนดเฟรมได้ที่ช่อง End และสามารถเริ่มเฟรมได้ที่ช่อง Begin สำหรับ Frame ที่ต้องการจะบันทึกข้อมูล ณ เวลานั้น หรือ Current Frame ให้เลือกที่ช่อ Position โดยใน Dialog นี้จะมีปุ่ม Button สำหรับทดลอง Play Pause Stop Record หรือ Delete Animation ก่อนทำการบันทึกลงไฟล์ด้วย สำหรับช่อง Record Node เป็นช่องสำหรับแสดง Frame ใน position ที่เราได้ทำการ Animation เอาไว้ทั้งหมด



รูปที่ 3.11 แสดงภาพ Dialog box File Management

สำหรับ การ Save Load หรือ Import File นั้น จะเป็นการบันทึกการเคลื่อนไหวของแต่ละเฟรมที่เราได้ใส่ข้อมูลเข้าไปทุกเฟรม เพื่อนำมาแสดงผลภายหลังได้ สำหรับปุ่ม Save ให้ทำการบันทึก Animation ในช่อง Animation Control ก่อน แล้วทำการกำหนดชื่อไฟล์แล้วจึงกด Save หรือถ้าต้องการเลือกไฟล์ที่เคยบันทึกไว้แล้ว ให้เลือก Load ส่วนการ Import สำหรับนำเฟรมที่ได้บันทึกไว้แล้วมาต่อกับไฟล์อื่นๆ ตามรูปที่ 3.11

สำหรับความสามารถเพิ่มเติมในการทำงานของโปรแกรม ที่นอกเหนือจากการเคลื่อนไหวตาม Input ข้อมูลที่ป้อนเข้าไปแล้ว ยังสามารถบันทึกข้อมูลเป็นไฟล์ โดยจะบันทึกเป็น Frame แล้วสามารถจะนำมาแสดงผลในภายหลังได้อีกด้วย

ในส่วนของการทำงานของตัวโปรแกรม จะมีคีย์ลัด (Hot Key) สำหรับใช้ในการแสดงผลต่างๆ ดังนี้

กด Tab	แสดงผล	เปิด – ปิด Dialog box ทั้ง 5 ส่วน
กด Ctrl + A	แสดงผล	แสดงฉากหลัง/ปิดฉากหลัง
กด ลูกศร ซ้ายขวา	แสดงผล	เปลี่ยนฉากหลัง
กด Ctrl + Mouse ซ้าย	แสดงผล	หมุนวัตถุ
กด Ctrl + Mouse กลาง	แสดงผล	ซูมภาพวัตถุ
กด Ctrl + Mouse ขวา	แสดงผล	แพน (Pan) กล้อง
กด Ctrl + C	แสดงผล	เปิด/ปิด Coordinate
กด Ctrl + D	แสดงผล	Default View
กด Ctrl + G	แสดงผล	Ground On/off

3.2 การพัฒนาโครงงาน

การสร้าง Application ทั้งหมดนี้ เริ่มจากการสร้างโปรแกรมตามหลักมาตรฐานสากลในการสร้างตัวการ์ตูนมนุษย์ให้สมจริง โดยมีโครงสร้างโปรแกรม แบบ Hierarchy ด้วยภาษา C++ โดยทำการลง DirectX 9.0 ไปด้วยเพื่อช่วยในการแสดงผลสามมิติในภาษา C ต่อมาสร้างในส่วนของตัวการ์ตูน 3 มิติ ให้มีจำนวน Node เท่ากับโหนด ที่กำหนดไว้ในโปรแกรมคือ 17 ส่วนดังกล่าวจากโปรแกรมสร้างภาพสามมิติ เช่น 3D Studio Max, Osca, Maya เป็นต้น โดยทางผู้จัดทำได้เลือกโปรแกรม Maya Version 4.5 มาใช้งาน เมื่อสร้างส่วนประกอบของตัวการ์ตูนสามมิติรูปคนสำเร็จทั้ง 17 ส่วน และแยกเป็น 17 ไฟล์ย่อยๆ แล้ว ทำการ Export File .ma ที่ได้จาก Maya มาเป็น File .x ซึ่งสามารถแสดงผลแบบสามมิติบนภาษา C++ ที่มี DirectX เป็นตัวช่วยจัดการแสดงผลได้

หลังจากได้ไฟล์ที่แปลงเป็น .x ทั้งหมด 17 ไฟล์แล้ว ทำการ run โปรแกรมโดยเปิดไฟล์ที่สร้างไว้ทั้งหมด แล้วทำการ Input ข้อมูลในการ Translate ตำแหน่งของแต่ละชิ้นส่วนลงไป ก็จะได้รูปโมเดลสามมิติรูปตัวการ์ตูนมนุษย์ ที่เราสร้างขึ้น สำหรับการเคลื่อนไหวส่วนต่างๆ ของร่างกาย ก็เพียงแต่ใส่ค่าการ Rotate เข้าไปตามส่วนต่างๆ โมเดลสามมิติของเราก็สามารถแสดงผลการเคลื่อนไหวแบบ Real-Time ได้แล้ว

เพียงแต่ใส่ค่าการ Rotate เข้าไปตามส่วนต่างๆ โมเดลสามมิติของเราก็สามารถแสดงผลการเคลื่อนไหวแบบ Real-Time ได้แล้ว

3.3 Class หลักที่ใช้ในตัวโปรแกรม

Class Human มีการ Define Class ตามหลักโครงสร้างลำดับชั้น VRML ดังต่อไปนี้

Source Code File

```
#include "CHuman.h"
#define HEAD    0
#define NECK    1
#define BODYT   2
#define BODYB   3
#define HIP     4
#define ARMLT   5
#define ARMLB   6
#define HANDL   7
#define LEGLT   11
#define LEGLB   12
#define FOOTL   13
#define ARMRT   8
#define ARMRB   9
#define HANDR   10
#define LEGRT   14
#define LEGRB   15
#define FOOTR   16


CHuman::CHuman(
                CMyD3DMesh *m_pHead,
                CMyD3DMesh *m_pNeck,
                CMyD3DMesh *m_pBodyT,
                CMyD3DMesh *m_pBodyB,
                CMyD3DMesh *m_pHip,
                CMyD3DMesh *m_pArmLT,
                CMyD3DMesh *m_pArmLB,
                CMyD3DMesh *m_pHandL,
                CMyD3DMesh *m_pLegLT,
                CMyD3DMesh *m_pLegLB,
                CMyD3DMesh *m_pFootL,
```

```
                    CMyD3DMesh *m_pArmRT,
                    CMyD3DMesh *m_pArmRB,
                    CMyD3DMesh *m_pHandR,
                    CMyD3DMesh *m_pLegRT,
                    CMyD3DMesh *m_pLegRB,
                    CMyD3DMesh *m_pFootR,
                    LPDIRECT3DDEVICE9 m_pd3dDeviceBuffer)
{

InitBoneComponant(17);
//Set Bone each part name
        //This steb is not important for CLASS MyD3DBone work
        //but usefull for work with animation tool
         SetComponantName(HEAD,"Head");
         SetComponantName(NECK,"Neck");
         SetComponantName(BODYT,"Body Top");
         SetComponantName(BODYB,"Body Bottom");
         SetComponantName(HIP,"Hip");
         SetComponantName(ARMLT,"Arm Left Top");
         SetComponantName(ARMLB,"Arm Left Bottom");
         SetComponantName(HANDL,"Hand Left");
         SetComponantName(LEGLT,"Leg Left Top");
         SetComponantName(LEGLB,"Leg Left Bottom");
         SetComponantName(FOOTL,"Foot Left");
         SetComponantName(ARMRT,"Arm Right Top");
         SetComponantName(ARMRB,"Arm Right Bottom");
         SetComponantName(HANDR,"Hand Right");
         SetComponantName(LEGRT,"Leg Right Top");
         SetComponantName(LEGRB,"Leg Right Bottom");
         SetComponantName(FOOTR,"Foot Right");
         //Set whole bone object 's name
         SetComponantName(m_iBoneCount,"Whole Human Body");


//Set Pointer for Mesh data
         SetComponantMeshIndex(HEAD,m_pHead);
         SetComponantMeshIndex(NECK,m_pNeck);
         SetComponantMeshIndex(BODYT,m_pBodyT);
         SetComponantMeshIndex(BODYB,m_pBodyB);
         SetComponantMeshIndex(HIP,m_pHip);
```

```
        SetComponantMeshIndex(ARMLT,m_pArmLT);
        SetComponantMeshIndex(ARMLB,m_pArmLB);
        SetComponantMeshIndex(HANDL,m_pHandL);
        SetComponantMeshIndex(LEGLT,m_pLegLT);
        SetComponantMeshIndex(LEGLB,m_pLegLB);
        SetComponantMeshIndex(FOOTL,m_pFootL);
        SetComponantMeshIndex(ARMRT,m_pArmRT);
        SetComponantMeshIndex(ARMRB,m_pArmRB);
        SetComponantMeshIndex(HANDR,m_pHandR);
        SetComponantMeshIndex(LEGRT,m_pLegRT);
        SetComponantMeshIndex(LEGRB,m_pLegRB);
        SetComponantMeshIndex(FOOTR,m_pFootR);

//Set Inverse Mesh
        SetComponantInverse(ARMLT, TRUE, FALSE, FALSE);
        SetComponantInverse(ARMLB, TRUE, FALSE, FALSE);
        SetComponantInverse(HANDL, TRUE, FALSE, FALSE);
        SetComponantInverse(LEGLT, TRUE, FALSE, FALSE);
        SetComponantInverse(LEGLB, TRUE, FALSE, FALSE);
        SetComponantInverse(FOOTL, TRUE, FALSE, FALSE);

//Set Matrix Constant
        SetComponantVector(HEAD, 0.0f, 0.5f, 0.0f );
        SetComponantVector(NECK, 0.0f, 2.5f, 0.0f );
        SetComponantVector(BODYT, 0.0f, 0.0f, 0.0f );
        SetComponantVector(BODYB, 0.0f, 0.0f, 0.0f );
        SetComponantVector(HIP, 0.0f, 0.0f, 0.0f );
        SetComponantVector(ARMLT, -1.2f, 2.0f, 0.0f );
        SetComponantVector(ARMLB, 2.0f, 0.0f, 0.0f );
        SetComponantVector(HANDL, 1.7f, 0.0f, 0.0f );
        SetComponantVector(LEGLT, -0.5f, -0.9f, 0.2f );
        SetComponantVector(LEGLB, 0.2f, -2.4f, -0.08f );
        SetComponantVector(FOOTL, 0.15f, -3.0f, -0.15f );
        SetComponantVector(ARMRT, 1.2f, 2.0f, 0.0f );
        SetComponantVector(ARMRB, 2.0f, 0.0f, 0.0f );
        SetComponantVector(HANDR, 1.7f, 0.0f, 0.0f );
        SetComponantVector(LEGRT,0.5f, -0.9f, 0.2f  );
        SetComponantVector(LEGRB, 0.2f, -2.4f, -0.08f  );
        SetComponantVector(FOOTR, 0.15f, -3.0f, -0.15f );
```

```
//Set juntion Bone
        SetComponantJunction(HEAD,NECK);
        SetComponantJunction(NECK,BODYT);
        SetComponantJunction(BODYT,BODYB);
        SetComponantJunction(BODYB,ROOT_BONE);
        SetComponantJunction(HIP,ROOT_BONE);
        SetComponantJunction(ARMLT,BODYT);
        SetComponantJunction(ARMLB,ARMLT);
        SetComponantJunction(HANDL,ARMLB);
        SetComponantJunction(LEGLT,HIP);
        SetComponantJunction(LEGLB,LEGLT);
        SetComponantJunction(FOOTL,LEGLB);
        SetComponantJunction(ARMRT,BODYT);
        SetComponantJunction(ARMRB,ARMRT);
        SetComponantJunction(HANDR,ARMRB);
        SetComponantJunction(LEGRT,HIP);
        SetComponantJunction(LEGRB,LEGRT);
        SetComponantJunction(FOOTR,LEGRB);

//Set Initial Transformation Value
        SetRotateX(0);
        SetRotateY(0);
        SetRotateZ(0);
        SetScale(1.0f,1.0f,1.0f);
        SetTranslate(0,0,0);

        InitDeviceObjects( m_pd3dDeviceBuffer );
}

CHuman::~CHuman()
```

3.4 การทดลอง

เนื่องจากเรายังไม่สามารถทดลองในระบบจริงได้ เพราะระบบทั้งหมดยังไม่เสร็จดี ดังนั้น จึงแยกทดลองเฉพาะส่วนการแสดงผลการเคลื่อนไหวของโมเดลสามมิติ โดยการสร้าง Dialog Box เอาไว้สำหรับ Input ค่าของข้อมูลที่ต้องการแสดงผล จะสังเกตได้ว่า เมื่อ Input ข้อมูลเข้าไป โมเดลสามมิติของเราก็จะเคลื่อนไหวแสดงผลทันที แบบ Real-Time

3.3.1 โปรแกรมและอุปกรณ์ที่ใช้ในการทดลอง

โปรแกรม

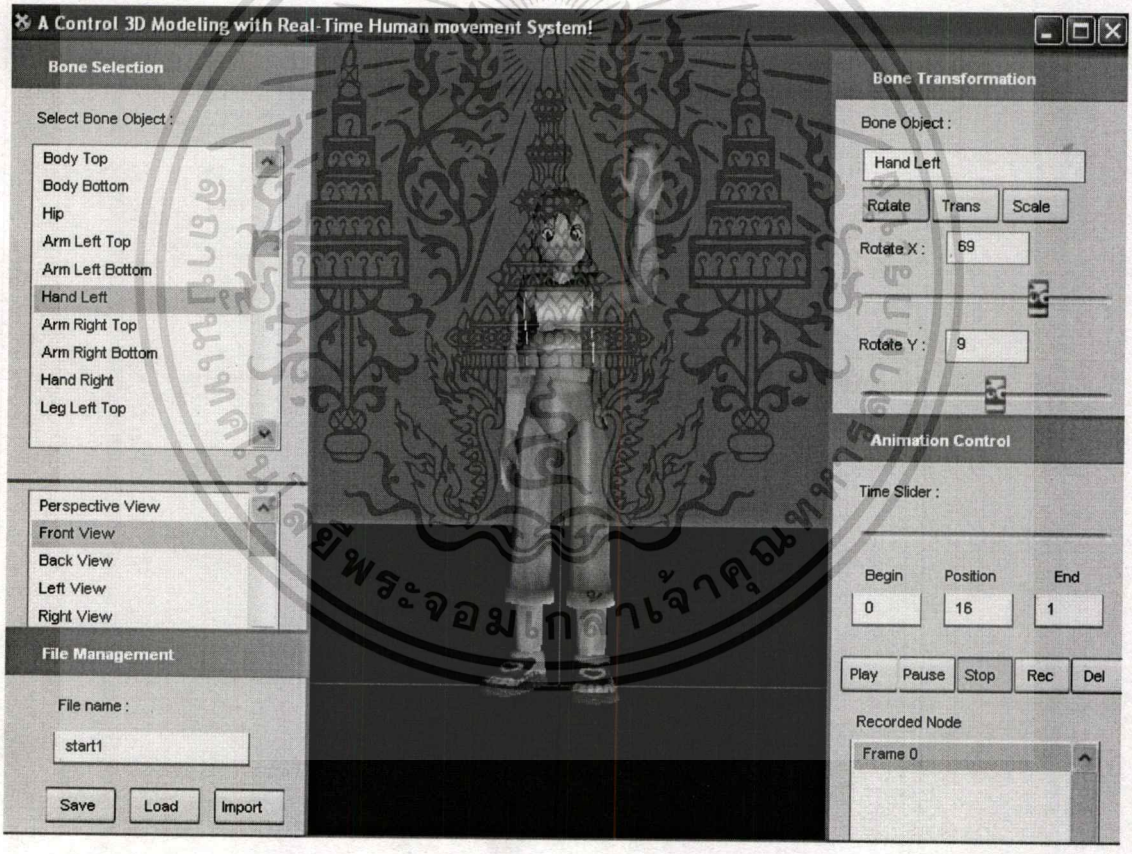
Window XP service Pack 2, Maya v 4.5, Microsoft Visual Studio 6.0 Enterprise Edition, Microsoft DirectX 9.0 SDK, Adobe Photoshop 7.0

เครื่องคอมพิวเตอร์

Pentium 4 Processor 1.8 GHz, 1024 MB DDR-RAM, 80 GB Harddisk ATA 5200 RPM, RADEON9600 64MB Soundcard, CD-ROM 40X

3.3.2 ขั้นตอนการทดลอง

เมื่อเริ่มเปิดโปรแกรม จะมีหน้าตาของ Application ดังนี้

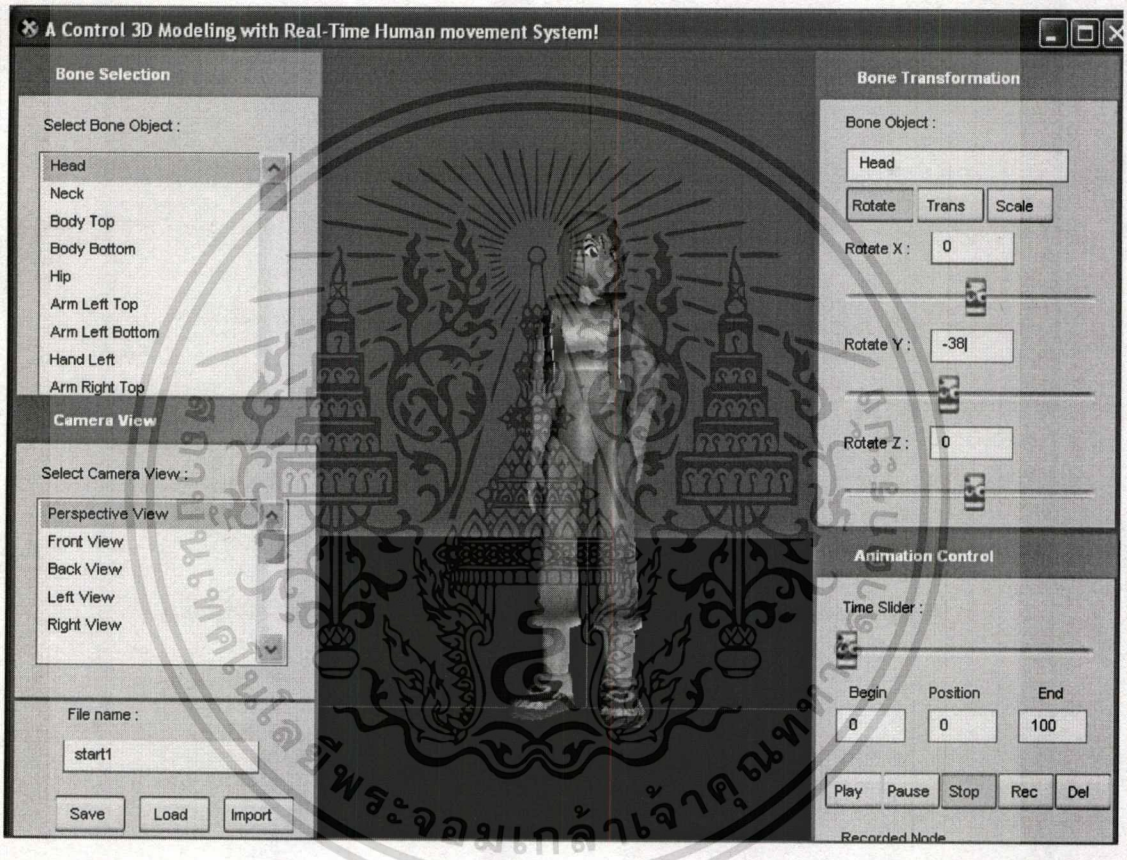


รูปที่ 3.12 แสดงหน้าจอส่วนประกอบของโปรแกรม

จากรูปที่ 3.12 เป็นการนำไฟล์ที่ได้ Record ไว้แล้วชื่อ Start1 มาแสดงผล โดยจะเป็นรูปตัวโมเดลสามมิติผู้หญิงทำการยกมือซ้ายขึ้น

เราสามารถเปลี่ยนมุมมอง หรือใส่ฉากหลังให้สวยงามได้โดยกด Ctrl + A และสำหรับตัวโปรแกรมเองนั้น จะทำการ Random ตัวการ์ตูนออกมา ซึ่งในการเปิดแต่ละครั้ง จะมีตัวการ์ตูนไม่ซ้ำกันอยู่ 3 รูปแบบด้วย

ทดลองใส่ค่าเข้าไปโดยทำการเลือก Node จากช่อง Bone Selection แล้วทำการเปลี่ยนค่าในช่อง Bone Transformation โดยเลือก rotate แล้วเลื่อน Slide bar แกน x, y, z สังเกตผลที่ได้ว่า ตัวการ์ตูน จะมีการเคลื่อนไหวตามค่าของ Slide bar ที่เราใส่เข้าไป เป็นต้น โดยรูปที่ 3.13 จะเป็นการแสดงการปรับค่าการหมุน (Rotate) ที่ส่วนของศีรษะ (Head) ในแนวแกน Y แล้วศีรษะเคลื่อนไหวตามค่าการ Rotate ที่ใส่เข้าไป คือ -38 องศา



รูปที่ 3.13 แสดงการปรับค่าเพื่อให้โมเดลเคลื่อนไหวในส่วนที่เลือก

3.5 สรุปผลการทดลอง

ระบบนี้สามารถทำงานได้อย่างถูกต้อง ตรงตาม Input ที่ใส่เข้าไป ทั้งการ Translate Rotate แ ละ scaling โดยถ้านำไปรวมกับส่วนอื่นๆ ทั้งหมดแล้ว จะสามารถ แสดงผลการเคลื่อนไหวได้อย่างสวยงาม เสมือนจริงและมีประสิทธิภาพ ซึ่งทั้งนี้ในระบบจริงจะต้องคำนึงถึงค่า Input ที่ผิดพลาดที่รับมาจากการคำนวณในส่วนการ Calibration อีกด้วย ซึ่งถ้าได้รับข้อมูลมาผิดพลาด ก็จะทำให้การแสดงผลผิดพลาดเช่นเดียวกัน

บทที่ 4
บทสรุปและข้อเสนอแนะ

4.1 บทสรุปของโครงการพัฒนาระบบงาน

โครงงานนี้ได้นำเสนอวิธีการทำโปรแกรม Application ซึ่งจะนำไปประยุกต์ใช้กับการทำภาพเคลื่อนไหวแบบสามมิติของแบบจำลองของร่างกายมนุษย์ โดยใช้มาตรฐานสากลที่นิยมกันมากในปัจจุบันคือ VRML มาตรฐานนี้จะทำให้ภาพที่สร้างออกมาสมจริง ทั้งยังไม่ขึ้นกับรูปแบบเบราเซอร์ (Browser) แอพพลิเคชัน (Application) หรือ แพลตฟอร์ม (platform) ใดๆ ก็ตาม จึงเกิดความยืดหยุ่นในการใช้งานอย่างสูง และสามารถนำไปพัฒนาเพิ่มเติมได้อีกด้วย

มาตรฐานสำคัญที่ใช้ในการสร้างรูปแบบจำลองมนุษย์และตัวการ์ตูนหรือ VRML จะมีการ Define โหนดแต่ละโหนด ซึ่งประกอบด้วย Joint Node, Segment Node, Site, View Point และ Humanoid Body และค่าแต่ละค่าที่จะต้องใส่เข้าไปในโหนดนั้นๆ เพื่อให้การเคลื่อนไหวของภาพสมจริงที่สุด โดยสำหรับตัวโปรแกรม จะมีการเก็บค่าสำคัญต่างๆ ในโหนดนั้นๆ คือ จุดเชื่อมต่อหรือ Joint Node คือ ข้อต่างๆ และ Segment Node คือ แขน ขา มือ ส่วนต่างๆ ที่ใช้ในการเชื่อมต่อกันเป็นรูปร่างมนุษย์ สำหรับแต่ละโหนดจะมีการบันทึกค่า Translate Rotate และ Scaling เพื่อใช้ในการบันทึกค่าไว้แสดงผลทีหลังได้ด้วย

สำหรับการนำไปพัฒนาต่อนั้น จะนำแอพพลิเคชันที่ได้ ไปประยุกต์ใช้กับระบบ Real-time 3D tracking and Rendering ต่อไป โดยจะมีผู้พัฒนาส่วนการประมวลผลผ่านกล้องเพื่อให้ได้ข้อมูลออกมาโดยใช้ภาษา C++ เช่นเดียวกัน ทำให้สามารถนำมารวมกันได้ง่ายอีกด้วย

4.2 ปัญหาและข้อจำกัดในการทำงานของระบบ

ในการพัฒนาระบบและออกแบบระบบช่วยตรวจจับการเคลื่อนไหวของร่างกายและนำข้อมูลที่ได้มาแปลงเป็นการเคลื่อนไหวของโมเดล 3 มิติรูปมนุษย์และตัวการ์ตูนนี้ มีปัญหาและข้อจำกัดในการพัฒนาอยู่หลายอย่าง เนื่องจากปัจจัยดังต่อไปนี้

4.2.1 ส่วนของการสร้างโมเดลสามมิติ จะต้องสร้างโมเดลแบบ Low Polygon หรือ เป็นโมเดลที่มีความละเอียดให้น้อยที่สุดเพราะเราต้องการการทำงานแบบ Real-Time ซึ่ง ถ้าเราทำโมเดลสามมิติเป็นแบบ High Polygon จะทำให้การวาดภาพ (Rendering) ในโหมด สามมิติช้าลง อาจทำให้ระบบไม่สามารถโต้ตอบกับผู้ใช้ได้ทันที หรือ อาจไม่ Real-Time ได้

4.2.2 สำหรับส่วนของการ Export File .x จาก Maya นั้น ยังมีข้อจำกัดบางอย่างซึ่งอาจทำให้ได้ภาพออกมาไม่เหมือนกับตอนที่อยู่ในตัวโปรแกรม Maya

4.2.3 ในส่วนของการแสดงผลแบบสามมิติ จำเป็นต้องใช้เครื่องที่มีประสิทธิภาพสูงๆ มี Monitor Card อย่างต่ำ 32 bit ขึ้นไป

4.2.4 ถ้าระบบรับ Input ข้อมูลที่ผิดพลาดเข้าไป การแสดงผลก็จะผิดพลาดตามไปด้วย

4.3 ข้อเสนอแนะในการพัฒนาระบบในอนาคต

4.3.1 เนื่องด้วยความน่าสนใจของระบบอยู่ที่รูปภาพที่แสดงออกมาสวยงาม ดังนั้นอนาคตภายหน้า น่าจะมีการพัฒนาการสร้างโมเดลสามมิติให้สวยและสมจริงกว่าที่เป็นอยู่ และน่าจะมี Feature เพิ่มในระบบ เช่น การรับค่าจาก User แล้วให้ระบบสร้าง Model เลียนแบบ User ขึ้นมา เช่น User เป็นผู้ชาย ใส่เสื้อสีเหลือง ระบบก็จะรับค่าผู้ชาย และเสื้อสีเหลืองเข้ามาประมวลผลแล้วแสดงตุ๊กตาแบบคล้ายกับ User ได้ เป็นต้น

4.3.2 ควรทำส่วนที่เป็นการดัก Error ของข้อมูลผิดพลาดที่ได้รับมาจากการ Calibrate กล้องเพื่อหาตำแหน่งส่วนของร่างกายต่างๆ เช่นอาจกำหนดการ rotate ของข้อมือไว้ว่าไม่ให้เกิน 180 องศา เป็นต้น

บรรณานุกรม

Creative Master, 2002 **3D Animation** [Online].Available:
http://www.netdesign.ac.th/3d_web/overview.html.

Hearn Donald, Baker Pauline M. 1999. **Computer Graphic** Second Edition.

Hilton, A. 2000a. **University of Surrey : seeing in 3I** [Online].Available:
http://maths.surrey.ac.uk/degrees/Exams/ExamPapers2000.

Hilton, A. 2000b. **Towards Model-Based Capture of a Persons Shape, Appearance and Motion** [Online].Available:
http://csdl.computer.org/comp/proceedings/mpeople/1999/0362/00/03620037abs.htm.

Humanoid Animation Working Group, 2001. **Specification for a standard VRML Humanoid Version 1.0** [Online].Available:
http://H-Anim.org/Specifications/H-Anim1.0/.

Lam, L. 1997. **Humanoid Animation Project: How it works** [Online].Available:
http://www.dlsproductions.com/hanim/tech.html.

Lead Animator, 2002. **Maya5 Intensive** [Online].Available:
http://www.netdesign.ac.th/maya_web/overview.html.

MSDN, 2003. **Directx9** [Online].Available:
http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/directx9_c/directx/graphics/programmingguide/fixedfunction/viewportsclipping/viewingfrustum.asp.

Takanishi, 2001. **Detail of the Sample Model** [Online].Available:
http://www.is.aist.go.jp/humanoid/openhrp/English/sample.html.

Technology Promotion Asociate, 2004. **What is VRML** [Online].Available:
http://www.tpa.or.th/newtpa/forum/viewforum.php?forum=10&378.

ประวัติผู้เขียน

ชื่อ นามสกุล	นางสาวอลิสา สิขัณฑกนาค
เกิดวันที่	27 มีนาคม 2524
ที่อยู่	19/2 หมู่ 3 ซอย สุขสวัสดิ์ ถนน สุวินทวงศ์ ตำบล คลองวังตะเคียน อำเภอเมือง จังหวัด ฉะเชิงเทรา 24000
สถานที่เกิด	อำเภอ ศรีราชา จังหวัด ชลบุรี
การศึกษา	วิทยาศาสตร์ วิทยาการคอมพิวเตอร์ มหาวิทยาลัย ศรีนครินทรวิโรฒ ประสานมิตร